

LEVEL

(Handwritten signature)

NSWC TR 80-18

AD A084022

PROGRAM DESIGN LANGUAGE ARCHITECTURE SPECIFICATION FOR THE AN/UYK-7 CENTRAL PROCESSOR

by
ALAN HYNSON
JAY MEYERS
CHARLES NAPLES
Strategic Systems Department

DTIC
ELECTE
MAY 9 1980

FEBRUARY 1980

Approved for public release; distribution unlimited.



NAVAL SURFACE WEAPONS CENTER

Dahlgren, Virginia 22448

Silver Spring, Maryland 20910

JDC FILE COPY

80 5 8 006

NAVAL SURFACE WEAPONS CENTER
Dahlgren, Virginia 22448

Paul L. Anderson, Capt., USN
Commander

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 14 NSWC TR-1-18	2. GOVT ACCESSION NO. AD-A084 021	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PROGRAM DESIGN LANGUAGE ARCHITECTURE SPECIFICATION OF THE AN/UYK-7 CENTRAL PROCESSOR (for) <u>Central</u>		5. TYPE OF REPORT & PERIOD COVERED Final Repts
7. AUTHOR(s) Alan J. Hynson Jay C. Meyers Charles J. Naples		8. CONTRACT OR GRANT NUMBER(s) 12. 3006
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Surface Weapons Center (K74) Dahlgren, VA 22448		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NIF
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Surface Weapons Center (K74) Dahlgren, VA 22448		12. REPORT DATE February 1980
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		13. NUMBER OF PAGES 203
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		15. SECURITY CLASS. (of this report) Unclassified
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		16. DECLASSIFICATION/DOWNGRADING SCHEDULE
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Program Design Language (PDL); emulation, computer architecture, microprogramming, AN/UYK-7, central processor (CP)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Univac AN/UYK-7 computer is the Navy's standard mainframe computer for medium and large scale applications. This report presents a Program Design Language (PDL) description of the AN/UYK-7 Central Processor (CP) architecture and is intended for use as a reference document. This PDL was used as a design specification for a microprogrammed emulation of the AN/UYK-7 Central Processor. This emulation has been (continued)		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
5 N 0102-LF-014-6601UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

411567

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

validated through the successful execution of the ~~Univac-supplied~~ AN/UYK-7 Central Processor (CP) diagnostics and by an independent testing effort.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

This document contains a comprehensive description of the central processor architecture of the Navy Standard AN/UYK-7 computer. The architecture was described using a Program Design Language (PDL) supplied by Caine, Farber, and Gordon, Inc. As far as can be determined this document contains the most accurate description of the central processor of the AN/UYK-7 computer published to date.

The PDL was used in the development of a microprogrammed AN/UYK-7 central processor emulation on a Nanodata QM-1 computer. This emulation has been validated through the successful execution of the UNIVAC-supplied AN/UYK-7 CP diagnostics and by an independent testing effort.

The authors gratefully acknowledge the efforts of Henry Walker (SPERRY UNIVAC) and Marc Hubbard (Gun Fire Control Systems Branch, Combat Systems Department) for providing assistance in clarifying ambiguous UNIVAC documentation and verification of architectural questions on actual AN/UYK-7 hardware and Helen Fletcher for assistance in the preparation of this document.

This report was prepared in the Programming Systems Branch of the Computer Programming Division and reviewed by Hermon W. Thombs, Head, Programming Systems Branch.

Released by:

R. T. Ryland, Jr.

R. T. RYLAND, JR., Head
Strategic Systems Department

Accession For	
NTIS	☒
DDC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution/ _____	
Availability Codes	
Dist	Avail and/or special
A	

TABLE OF CONTENTS

AN/UYK-7 CP SEQUENCES	2
CP MAIN LOOP	3
I_SEQUENCE	4
INTERRUPT_SCAN	5
DECODE	6
DECODE_PHASE_TWO	7
INTERRUPT_SEQUENCE	8
REPEAT_SEQUENCE	9
CP SUPPORT ROUTINES LEVEL I	10
GENERATE_SYNCHRONOUS_INTERRUPT	11
GET_ISC	12
JUMP_ADDRESS	13
OP_READ	14
OP_STORE	15
CP SUPPORT ROUTINES LEVEL II	16
BPR_CHECK	17
IA_SEQUENCE	18
MEMORY_READ	19
SPR_CHECK	20
CP INSTRUCTION SUPPORT ROUTINES	21
CP/IOC_CLOCK_COMMUNICATIONS	22
DO_JUMP	23
GET_SHIFT_AMOUNT	24
HALF-WORD_TOGGLE	25
REPLACE_CHECK	26
REPLACE	27
UPDATE_REPLACE	28
SET_CD1	29
SET_CD2	30
CP UTILITY ROUTINES	31
ADD_S	32
GET_ARG	33
GET_ARG	34
GET_SREG	35
PUT_ARG	36
PUT_ARG	37
PUT_SPEG	38
FLOATING POINT SUBROUTINES	39
FLOATING_ADD_SUBTRACT_HEADER	40
FLOATING_OVERFLOW	41
FLOATING_NORMALIZE	42
FLOATING_ROUND	43
ROUND_UP	44
DIVIDE_COMPARE	45
FLOATING_POINT_END	46

CP INSTRUCTION SET

-OR	47
-SC	48
-MS	49
-ROR	50
-ALP	51
-LLP	52
-CMT	53
-RR	54
-SLP	55
-SSUM	56
-SDIF	57
-DS	58
-TSF	59
-DL	60
-DA	61
-DC	62
-LAMP	63
-FA	64
-FAN	65
-FM	66
-FD	67
-XS	68
-IPI	69
-AEI	70
-LIM	71
-IO	72
-IR	73
-RP	74
-LA	75
-LKB	76
-LOIF	77
-ANA	78
-AA	79
-LSUM	80
-LMA	81
-LM	82
-LB	83
-AB	84
-AMB	85
-SB	86
-SA	87
-SXB	88
-SMA	89
-SM	90
-BZ	91
-RA	92
-RI	93
-M	94
-N	95
DO_MULTIPLY	96
DO_DIVIDE	97
-D	98
-BC	99
-CHI	100
-C	101

-CL	102
-CM	103
-CG	104
-JEP	105
-DJZ	106
-DJM2	107
-F31	108
-LBJ	109
-JBM2	110
-JS	111
-JL	112
-J530	113
-J531	114
-J532	115
-J533	116
-LCT	117
-SCT	118
-MSC-60	119
-MLC-61	120
-MLC	121
-MLC	122
-MSF	123
-MDSF	124
-MCP	125
-MCP	126
-MCP	127
-MCP	128
-MH	129
-MD	130
-MRT	131
-DO_SORT	132
-MLB	133
-MC	134
-MCL	135
-MCM	136
-MCB	137
-MSM	138
-MSTC	139
-MPI	140
-M776	141
AN-UYN-7 CP EMULATION GLOSSARY	142
AN/UYN-7 CP EMULATION STATUS/CONTROL INDICATORS	143
STATUS/CONTROL INDICATORS PAGE 2	144
AN/UYN-7 CP ARCHITECTURE DESCRIPTORS	145
AN/UYN-7 CP MAINTENANCE PANEL SUPPORT	146
AN/UYN-7 CP EMULATION INTERNAL REGISTER SUPPORT	147
DISTRIBUTION	148
DISTRIBUTION LIST (GOVERNMENT INSTALLATIONS)	149
DISTRIBUTION LIST (GOVERNMENT INSTALLATIONS CONTINUED)	150
DISTRIBUTION LIST (PRIVATE INDUSTRY)	151
DISTRIBUTION LIST (LOCAL)	152
REFERENCES	153

NSWC

AN/UYK-7 ICPI
TABLE OF CONTENTS

14 DEC 79 PAGE 1.004

DATA INDEX	184
FLOW SEGMENT INDEX	199

AM/UYK-7 (CP)

14 DEC 79 PAGE 2

* AN/UYK-7 CP SEQUENCES *

CP MAIN LOOP FOR AN/UYK-7 CENTRAL PROCESSOR

```

14  * 1  DO FOREVER --LOOP THROUGH INSTRUCTION, INDIRECT ADDRESS, OPERAND & INTERRUPT SEQUENCES.
16  * 2  IE _STEP SET, THEN
17  * 3  SUSPEND LOOPING UNTIL RESTARTED
18  * 4  ENDIE
19  * 5  IE ASR(13,1) "CLASS II LOCKOUT SET" .OR.
20  * 6  ASR(11,1) "INTERUPT BASE REGISTERS SELECTED" .OP.
21  * 7  ASR(9,1) "MEMORY LOCKOUT INHIBIT SET", THEN
22  * 8  DISABLE SPP_CHECKS
23  * 9  ELSE
24  * 10  ENABLE SPP_CHECKS
25  * 11  ENDIE
26  * 12  IE NOT REPEAT_IN_PROGRESS, THEN
27  * 13  CLEAR MEMORY_STORE_INDICATOR --THIS INDICATOR IS SET BY INSTRUCTIONS
28  * 14  --THAT STORE INTO MEMORY. IT IS USED IN THE REPEAT TERMINATION LOGIC.
29  * 15  CALL I_SEQUENCE --FETCH INSTRUCTION, PERFORM BREAKPOINT & SPP CHECKS.
30  * 16  ENDIE
31  * 17  IE INTERRUPT_SCAN_INHIBIT CLEAR, THEN
32  * 18  CALL INTERRUPT_SCAN --CHECK FOR ASYNCHRONOUS INTERRUPTS.
33  * 19  ENDIE
34  * 20  CALL _DECODE --EXECUTE INSTRUCTION INDICATED BY UCFUNCTION DESIGNATORS).
35  * 21  IE REPEAT_IN_PROGRESS, THEN
36  * 22  CALL REPEAT_SEQUENCE --PERFORM REPEAT TERMINATION LOGIC.
37  * 23  ELSE
38  * 24  IE ASR(15,1) "LOWER HALF-WORD INSTRUCTION", THEN
39  * 25  SET INTERRUPT_SCAN_INHIBIT --DON'T ALLOW ASYNCHRONOUS INTERRUPTS
40  * 26  --BETWEEN HALF-WORD INSTRUCTIONS.
41  * 27  ELSE
42  * 28  CLEAR INTERRUPT_SCAN_INHIBIT
43  * 29  ENDIE
44  * 30  ENDIE
45  * 31  IE CP MONITOR CLOCK POSITIVE (I.E. >= 0), THEN
46  * 32  DECREMENT CP MONITOR CLOCK EACH 1/1024 SECOND
47  * 33  IE CP MONITOR CLOCK NEGATIVE, THEN
48  * 34  GENERATE CP MONITOR CLOCK INTERRUPT TO BE DETECTED BY INTERRUPT_SCAN
49  * 35  ENDIE
50  * 36  ENDIE
51  * 37  ENDDO
52  *
53  *
54  *

```

I_SEQUENCE

```

REF
PAGE
*****
56 * 1  .. MOVE THE NEXT INSTRUCTION FROM MEMORY INTO THE U REGISTER.
57 * 2  ..PERFORM APPROPRIATE BREAKPOINT AND SOFTWARE PROTECTION REGISTER (SPR) CHECKS.
58 * 3  32_REG(1) := P(0) - ASR(15,1)  ..THE PROPER DISPLACEMENT TO THE NEXT INSTRUCTION.
59 * 4  .. REFETCH U ON A LOWER HALF-WORD SINCE A SYNCHRONOUS INTERRUPT MAY
60 * 5  ..HAVE BEEN PROCESSED SINCE EXECUTION OF THE UPPER HALF-WORD INSTRUCTION.
61 * 6  32_REG(1) := 32_REG(1)  ..GET TEMPORARY COPY OF DISPLACEMENT.
62 * 7  CALL ADD_5 (PCS), 32_REG(1)  ..CONDITIONALLY ADD IN S REGISTER.
63 * 8  IE ASR(15,1)=0  ..NOT EXECUTING LOWER HALF OF INSTRUCTION", THEN
64 * 9  CALL BPR_CHECK (32_REG(1), INSTRUCTION ADDRESS)  ..CHECK FOR BREAKPOINT.
65 * 10  ENDIE
66 * 11  CALL SPR_CHECK (PCS), INSTRUCTION EXECUTE, "P=0, 32_REG(1)
67 * 12  CALL MEMORY_READ (32_REG(1), U, PCS), INSTRUCTION, "P=0)
68 * 13  IE ASR(15,1) SET "EXECUTING LOWER HALF-WORD INSTRUCTION", THEN
69 * 14  UU := UU  ..SHIFT LOWER HALF-WORD INSTRUCTION
70 * 15  ENDIE
71 * 16  CLEAR EXECUTE_REMOTE_IN_PROGRESS
72 * 17  CLEAR CHARACTER_ADDRESSING_OVERRIDE
73 * 18  CLEAR SPR_PRIVILEGED_INSTRUCTION  ..ASSUME NO SPECIAL PRIVILEGED INSTRUCTION CHECKS
74 * 19  PCC := 32_REG(1)+1  ..INCREMENT P TO NEXT INSTRUCTION ADDRESS.
75 * 20  DEFINE INSTRUCTION_FORMAT_INDICATOR  ..I.E. I, II, III, OR IV.
76 * 21  IE REPEAT_PENDING, THEN
77 * 22  IE INSTRUCTION_REPEATABLE (AS PER REPERTOIRE CARD), THEN
78 * 23  SET REPEAT_IN_PROGRESS
79 * 24  CLEAR REPEAT_PENDING
80 * 25  SET SPR_PRIVILEGED_INSTRUCTION  ..ALL REPEATED INSTRUCTIONS PRIVILEGED WHEN
81 * 26  ..SPR(16,1) SET "INTERRUPT OKS FOR INDIRECT ADDRESSING" AND INDIRECT ADDRESSING.
82 * 27  IE GO(UF)>3  "CMP REFERENCE INSTRUCTION", THEN
83 * 28  SET INTERRUPT_SCAN_INHIBIT  ..DON'T ALLOW DETECTION OF ASYNCHRONOUS INTERRUPTS
84 * 29  ..WHEN REPEATING A CBR REFERENCE INSTRUCTION.
85 * 30  ENDIE
86 * 31  ELSE
87 * 32  CLEAR REPEAT_PENDING
88 * 33  ABORT THE INSTRUCTION
89 * 34  ENDIE
90 * 35  RETURN
91 * 36
92 *
93 *
94 *
95 *
96 *
97 *
98 *
*****

```


NS=C AN/UYH-7 (CP)
AN/UYK-7 CP SEQUENCES

_DECODE

```

141 * 1  .. DECODE INSTRUCTION IN U BASED 1 U(F). SOME INSTRUCTIONS REQUIRE
142 * 2  ..SUBFUNCTION FIELDS (SEE DECODE..BASE_TWO).
143 * 3  IF UCS=7 .AND. ASR(B,1)=1 .AND. (.NOT. LOPP INSTRUCTION) .AND. TASK
144 * 4  MODE, THEN
145 * 5  CALL GENERATE_SYNCHRONOUS_INTERRUPT
146 * 6  (MP=-1, PRIVILEGED INSTRUCTION, DUMMY_CODE) ..ABORT INSTRUCTION.
147 * 7  ENDIF
148 * 8
149 * 9  DO CASE U(F) CORRESPONDING TO THE FOLLOWING TABLE
150 * 10
151 * 11  ILLEGAL, F01, F02, F03, ILLEGAL, F05, F06, F07,
152 * 12  _LA, _LXB, _LDIF, _ANA, _AA, _LSUP, _LNA, _LM,
153 * 13  _LB, _AB, _ANP, _SB, _SA, _SXB, _SNA, _SM,
154 * 14  ILLEGAL, ILLEGAL, _BZ_DS, _BZ_DS, _RA, _PI, _PAN, _PD,
155 * 15  _M, _D, _BC, _CHI, _C, _CL, _CM, _CG,
156 * 16  F50, _F51, F52, F53, _LCT, _LCI, _SCI,
157 * 17  HSC_60, HLC_61, _MLC, _MOLC, _MRZ, _MDPZ, _MPS, _MPS,
158 * 18  F70, F71, ILLEGAL, ILLEGAL, F74, ILLEGAL, ILLEGAL, F77
159 * 19  ENDDC
160 * 20
161 * 21  ILLEGAL
162 * 22  CALL GENERATE_SYNCHRONOUS_INTERRUPT
163 * 23  (MP=-1, CP ILLEGAL INSTRUCTION, DUMMY_CODE) ..ABORT INSTRUCTION.
164 * 24
165 * 25
166 * 26
167 * 27
168 * 28
169 * 29
170 * 30
171 * 31
172 * 32

```

NSWC AN/UYK-7 (CP)
AN/UYK-7 CP SEQUENCES

DECODE_PHASE_TWO

```

174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216

```

 1 NOTE: ----> ILLEGAL INSTRUCTION.
 2
 3 F01 : DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
 4 _OP, _SC, _NS, _XDR, _ALP, _LLP, _MLP, _LLPM
 5
 6 F02 : DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
 7 _CNT, _----, _XP, _XPL, _SLP, _SSUM, _SOIF, _OS
 8
 9 F03 : DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
 10 _ROR, _RSC, _RMS, _ROR, _RALP, _RLP, _RMLP, _TSF
 11
 12 F05 : DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
 13 _CL, _DA, _DAN, _DC, _LEMP, _----, _----
 14
 15 F06 : DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
 16 _FA, _FAM, _FM, _FO, _FAR, _FANR, _FMR, _FDP
 17
 18 F07 : DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
 19 FG70, _AET, _PEI, _LIP, _IO, _IO, _RP, _----
 20
 21 F070 : DO CASE U(A+2,1) CORRESPONDING TO THE FOLLOWING TABLE
 22 _RS, _IFI
 23
 24 F50 : DO CASE U(F3) CORRESPONDING TO THE FOLLOWING TABLE
 25 _JEP, _JOP, _DJZ, _DJNZ
 26
 27 F52 : DO CASE U(F3) CORRESPONDING TO THE FOLLOWING TABLE
 28 _LBJ, _JBNZ, _JS, _JL
 29
 30 F53 : DO CASE U(F3) CORRESPONDING TO THE FOLLOWING TABLE
 31 _J530, _J531, _J532, _J533
 32
 33 F70 : DO CASE U(F4) CORRESPONDING TO THE FOLLOWING TABLE
 34 _HSP, _HDSF, _MCP, _MDCP, _----, _----, _----
 35
 36 F71 : DO CASE U(F4) CORRESPONDING TO THE FOLLOWING TABLE
 37 _MOP, _MA, _MAN, _MICE, _----, _HAND, _----
 38
 39 F74 : DO CASE U(F4) CORRESPONDING TO THE FOLLOWING TABLE
 40 _MM, _MD, _MET, _MLB, _MC, _MCL, _MCM, _MCR
 41
 42 F77 : DO CASE U(F4) CORRESPONDING TO THE FOLLOWING TABLE
 43 _MSIM, _MSTC, _----, _HPI, _MAI, _M776, _----
 44

INTERRUPT_SEQUENCE (I_CLASS, _ISC)

```

218 * 1  ** P MODIFICATION PERFORMED PRIOR TO ENTRY.
219 * 2  **THERE IS NO PROVISION FOR POWER TOLERANCE INTERRUPTS (I.E.
220 * 3  **NEVER OCCUR, THUS NEVER HANDLED) IN THIS EMULATION.
221 * 4  ** (I) _CLASS - THE CLASS (LEVEL) OF INTERRUPT BEING PROCESSED.
222 * 5  ** (I) _ISC - INTERRUPT STATUS CODE ASSOCIATED WITH THIS INTERRUPT.
223 * 6  ** ASP -> CPM (0'139' + 4*_CLASS) --SAVE ACTIVE STATUS REGISTER.
224 * 7  ** _ISC -> CPM (0'136' + 4*_CLASS) --SAVE INTERRUPT STATUS CODE.
225 * 8  ** P -> CPM (0'137' + 4*_CLASS) --SAVE PROGRAM COUNTER.
226 * 9  ** DO CASE _CLASS OF
227 * 10  ** IV --MUST BE NON-POWER TOLERANCE (SEE COMMENT ABOVE).
228 * 11  ** ASP(19,20) := 0'2077600' --LEAVE CPID. SET STATE I, CLASS I, II, III LOCKOUTS.
229 * 12  **SET INTERRUPT S#0 SELECTS, MEMORY LOCKOUT INHIBIT, LBMP ENABLE AND BOOTSTRAP.
230 * 13  ** P(S) := 7; P(C) := 0 --FORCE MDR0 SWITCHER ACTIVATION.
231 * 14  ** IV
232 * 15  ** ASP(19,20) := ASR(19,20)+A.0'40200' --LEAVE CPID, CLASS I LOCKOUTS,
233 * 16  **AND BOOTSTRAP BITS UNCHANGED.
234 * 17  ** ASP(19,20) := ASR(19,20)+V.C'137400' --SET STATE II, CLASS II, III LOCKOUTS.
235 * 18  **SET INTERRUPT S#3 SELECTS, MEMORY LOCKOUT INHIBIT & LBMP ENABLE.
236 * 19  ** IE AUTO-RECOVERY SELECTED .AND. _ISC=B'001C' "ILLEGAL INSTRUCTION", THEN
237 * 20  ** ASR(7,13) := 1 --SET BOOTSTRAP MCODE.
238 * 21  ** P(S) := 7; P(C) := 1 + BOOTSTRAP_SWITCH_SETTING --II.E. 1,2,OR 3)
239 * 22  ** ELSE
240 * 23  ** P := CPM(0'144') --CLASS II ICW.
241 * 24  ** ENDIE
242 * 25  ** IV
243 * 26  ** ASP(19,20) := ASR(19,20)+A.0'60260' --LEAVE CPID, CLASS I & II LOCKOUT,
244 * 27  **AND BOOTSTRAP BITS UNCHANGED.
245 * 28  ** ASP(19,20) := ASR(19,20)+V.C'417400' --SET STATE III, CLASS III LOCKOUT.
246 * 29  **SET INTERRUPT S#0 SELECTS, MEMORY LOCKOUT INHIBIT & LBMP ENABLE.
247 * 30  ** P := CPM(0'150') --CLASS III ICW.
248 * 31  ** IV
249 * 32  ** ASP(19,20) := ASR(19,20)+A.0'70200' --LEAVE CPID, BOOTSTRAP BIT AND LOCKOUTS UNCHANGED.
250 * 33  ** ASP(19,20) := ASR(19,20)+V.0'207400' --SET STATE IV.
251 * 34  **SET INTERRUPT S#0 SELECTS, MEMORY LOCKOUT INHIBIT & LBMP ENABLE.
252 * 35  ** P := CPM(0'154') --CLASS IV ICW.
253 * 36  ** ENDDO --END CASE.
254 * 37  ** BELTEN --TO END OF MAIN LOOP (ABORTING INSTRUCTION).
255 *

```

REPEAT_SEQUENCE

```

*****
REF PAGE
263 * 1 .. PERFORM REPEAT TERMINATION LOGIC.
264 * 2 B(7) := B(7) - 1 ..DECREMENT REPEAT COUNTER.
265 * 3 B(8) := B(8) + REPEAT_SY ..INCREMENT B REGISTER WITH SY FIELD OF REPEAT INSTRUCTION
266 * 4 IE B(7) = 0, THEN
267 * 5 CLEAR REPEAT_IN_PROGRESS INDICATOR
268 * 6 ELSE ..CHECK REPEAT TERMINATION CONDITIONS (OTHER THAN B(7)=0).
269 * 7 IE (U(31,6)>>42 .AND. U(31,6)<50) .OR. (U(31,6)=0'03' .AND. U(22,3)=7) "TSF", THEN ..A
270 * 8 COMPARE INSTRUCTION.
271 * 9 DO CASE REPEAT_ARCA) OF
272 * 10 101 IF ASR(2,1)=0 "<0", THEN CLEAR REPEAT_IN_PROGRESS ENDF
273 * 11 111 IF ASR(2,1)=1 ">0", THEN CLEAR REPEAT_IN_PROGRESS ENDF
274 * 12 121 IF ASR(2,2)=0'01' ">0", THEN CLEAR REPEAT_IN_PROGRESS ENDF
275 * 13 131 IF ASR(1,1)=1 ">0", THEN CLEAR REPEAT_IN_PROGRESS ENDF
276 * 14 141 IF ASR(1,1)=0 "<0", THEN CLEAR REPEAT_IN_PROGRESS ENDF
277 * 15 151 IF ASR(2,1)=1 .OR. ASR(1,1)=0 "<0", THEN CLEAR REPEAT_IN_PROGRESS ENDF
278 * 16 161 IF ASR(0,1)=1 "OUT OF LIMITS", THEN CLEAR REPEAT_IN_PROGRESS ENDF
279 * 17 171 IF ASR(0,1)=0 "WITHIN LIMITS", THEN CLEAR REPEAT_IN_PROGRESS ENDF
280 * 18 ENDDO ..END CASE.
281 * 19 ELSE ..NOT A COMPARE INSTRUCTION.
282 * 20 DO CASE REPEAT_ARCA) OF
283 * 21 101 IF REPEAT_ACCUMULATOR <0, THEN CLEAR REPEAT_IN_PROGRESS ENDF
284 * 22 111 IF REPEAT_ACCUMULATOR =0, THEN CLEAR REPEAT_IN_PROGRESS ENDF
285 * 23 121 IF REPEAT_ACCUMULATOR >0, THEN CLEAR REPEAT_IN_PROGRESS ENDF
286 * 24 131 IF REPEAT_ACCUMULATOR <0, THEN CLEAR REPEAT_IN_PROGRESS ENDF
287 * 25 ... CASE 4 IMPLIES DO NOT TERMINATE
288 * 26 151 IF REPEAT_ACCUMULATOR EVEN PARITY .AND. MEMORY_STORE_INDICATOR, THEN CLEAR
289 * 27 REPEAT_IN_PROGRESS ENDF
290 * 28 161 IF REPEAT_ACCUMULATOR ODD PARITY .AND. MEMORY_STORE_INDICATOR, THEN CLEAR
291 * 29 REPEAT_IN_PROGRESS ENDF
292 * 30 ... CASE 7 IMPLIES DO NOT TERMINATE
293 * 31 ENDDO ..END CASE.
294 * 32 ENDF
295 * 33 IE .NOT. REPEAT_IN_PROGRESS, THEN
296 * 34 CLEAR INTERRUPT_SCAN_INHIBIT INDICATOR
297 * 35 ENDF
298 * 36 RETURN
299 *
300
301
302
303

```

MSHC

AN/UYK-7 (CP)

14 DEC 79 PAGE 10

*
* CP SUPPORT ROUTINES LEVEL I *
*

GENERATE_SYNCHRONOUS_INTERRUPT IP_MODIFICATION, _INTERRUPT, _CODE

```

306 1  ** GENERATE THE APPROPRIATE SYNCHRONOUS INTERRUPT, BUILD THE INTERRUPT STATUS CODE (ISC),
308 2  **AND MODIFY PROGRAM COUNTER FOR STORAGE AS INDICATED.
309 3  **CURRENT INSTRUCTION IS ADDED WHETHER OR NOT THE INTERRUPT IS ACTUALLY GENERATED.
310 4  ** (1) P_MODIFICATION -- P REGISTER MODIFICATION VALUE.
311 5  ** (2) _INTERRUPT -- INTERRUPT INFORMATION.
312 6  ** (3) _CODE -- INFORMATION FOR ISC GENERATION (MEMORY BANK, IOC#, ETC.).
313 7  IF _INTERRUPT IS A CLASS I, THEN
314 8  CLASS := 1 --CLASS INDICATOR FOR INTERRUPT SEQUENCE.
315 9  IE ASR(14,1) "CLASS I LOCKED OUT", THEN
316 10  RETURN --TO MAIN LOOP (LOCKED OUT IMPLIES NO INTERRUPT GENERATED).
317 11  ELSE
318 12  IE _INTERRUPT IS A CLASS II, THEN
319 13  CLASS := II --CLASS INDICATOR FOR INTERRUPT SEQUENCE.
320 14  IE ASR(13,1) "CLASS II LOCKED OUT", THEN
321 15  RETURN --TO MAIN LOOP (LOCKED OUT IMPLIES NO INTERRUPT GENERATED).
322 16  ELSE
323 17  ENDIE
324 18  ELSE **MUST BE CLASS IV (REI) SINCE THERE ARE NO CLASS III SYNCHRONOUS INTERRUPTS.
325 19  CLASS := IV --CLASS INDICATOR FOR INTERRUPT SEQUENCE.
326 20  **NOTE: THE CLASS IV INTERRUPT (REI) IS NEVER LOCKED OUT.
327 21  ENDIE
328 22  DO CASE _INTERRUPT OF
329 23  NOP MEMORY RESUME\ ISC := _CODE .COM. B'0000' --_CODE = MEMORY BANK NUMBER (0-15).
330 24  IOC COMMAND RESUME\ ISC := _CODE .COM. B'00000001' --_CODE = IOC NUMBER.
331 25  INSTRUCTION MEMORY RESUME\ ISC := _CODE .COM. B'0010' --_CODE = MEMORY BANK NUMBER (0-15).
332 26  IOC INTERRUPT CODE RESUME\ ISC := _CODE .COM. B'00000011' --_CODE = IOC NUMBER.
333 27  IFLOATING POINT ERROR\ ISC := B'0001'
334 28  VCP ILLEGAL INSTRUCTION ERROR\ ISC := B'0010'
335 29  VPRIVILEGED INSTRUCTION ERROR\ ISC := B'0011'
336 30  VOPERAND BREAKPOINT MATCH\ ISC := B'0101'
337 31  VOPERAND READ OR INDIRECT ADDRESSING\ ISC := B'0110'
338 32  VOPERAND LIMIT\ ISC := B'1001'
339 33  VOPERAND LIMIT\ ISC := B'1010'
340 34  VOPERAND LIMIT\ ISC := B'1011'
341 35  VOPERAND LIMIT\ ISC := B'1101'
342 36  VOPERAND LIMIT\ ISC := B'1110'
343 37  VREI\ ISC := _CODE --_CODE EQUIVALENT TO ISC FOR REI.
344 38  ENDDO --END CASE.
345 39  DO CASE --MODIFY PROGRAM COUNTER FOR INTERRUPT STORAGE.
346 40  REPEAT_PENDING\ P := P-1 --REEXECUTE REPEAT.
347 41  REPEAT_IN_PROGRESS\ P := P-2 --REEXECUTE REPEAT.
348 42  ELSE
349 43  P := P-P_MODIFICATION --SO PROPER RETURN CAN BE MADE.
350 44  ENDDO --END CASE.
351 45  CALL INTERRUPT_SEQUENCE (CLASS, ISC)
352 46  RETURN
353 47

```

GET_ISC (_CLASS, REF_CODE)

REF
PAGE

```
362 * 1 .. ACQUIRE INTERRUPT STATUS CODE (ISC) FROM APPROPRIATE IOC AND ADD
363 * 2 ..IOC NUMBER IN BITS 9 & 8 BEFORE RETURNING (ISC) TO CALLER. SHOULD
364 * 3 ..THE IOC RETURN AN INVALID ISC, THIS FACT MUST BE CONVEYED TO THE CALLER.
365 * 4 .. (1) _CLASS -- INTERRUPT LEVEL (1 OR 3) OF CONCERN.
366 * 5 .. (2) _CODE -- ISC RECEIVED FROM IOC AND RETURNED TO CALLER.
367 * 6 DETERMINE IOC_# CAUSING INTERRUPT
368 * 7 RECOGNIZE RECEIPT OF INTERRUPT BY MAKING AN IOC REQUEST ON IOC_#
369 * 8 SEND INTERRUPT CLASS DESIGNATOR (_CLASS) VIA O_BUS
370 * 9 WAIT UNTIL IOC ACQUIRES CLASS INDICATOR
371 * 10 RECEIVE ISC FROM IOC_# ACROSS O_BUS
372 * 11 REACTIVATE CLASS(_CLASS) INTERRUPT LINE
373 * 12 _CODE := IOC_# .CON. ISC ..2-BIT IOC # CONCATENATED WITH 8-BIT ISC (POSSIBLY INVALID).
374 * 13 RETURN
375 *
```

JUMP_ADDRESS(M_DISPLACEMENT, REF_OPERAND_S, REF_OPERAND_DISPLACEMENT)

```

REF
PAGE
*****
1  .. CALLED BY JUMP TYPE INSTRUCTIONS. FINDS THE "JUMP TO" ADDRESS AND
2  .. PERFORMS BREAKPOINT AND SPR CHECKS.
3  .. (1) M_DISPLACEMENT - 0/1 FOR OPI/OP2 SEQUENCE RESPECTIVELY.
4  .. (2) OPERAND_S - THE BASE REGISTER PORTION OF THE ADDRESS.
5  .. (3) OPERAND_DISPLACEMENT - THE 16-BIT DISPLACEMENT PORTION OF THE ADDRESS.
6  IE U(1) "INDIRECT ADDRESSING", THEN ..DO CASCADING.
7      CALL IA_SEQUENCE (OPERAND_S, OPERAND_DISPLACEMENT, DUM_C, DUM_C1, DUM_P, DUM_MASK)
8      ..DUMMY C, P, MASK SINCE CHARACTER ADDRESSING HAS NO MEANING FOR JUMP INSTRUCTIONS.
9      OPERAND_DISPLACEMENT := OPERAND_DISPLACEMENT + M_DISPLACEMENT
10     ELSE
11         OPERAND_DISPLACEMENT := U(CY) + 8(U(CB))(15) + M_DISPLACEMENT
12         OPERAND_S := U(S)
13     ENDIE
14 32_REG(1) := OPERAND_DISPLACEMENT ..GET COPY FOR USE IN CHECKING.
15 CALL SPR_CHECK (OPERAND_S, INSTRUCTION_EXECUTE, "P-1", OPERAND_DISPLACEMENT) ..ECP(66).
16 CALL ADD_S (OPERAND_S, 32_REG(1)) ..COMPUTE FINAL (ABSOLUTE) ADDRESS.
17 CALL MEMORY_READ (32_REG(1), DUMMY, OPERAND_S, "P-1") ..CHECK FOR OPERAND MEMORY RESUME.
18 CALL SPR_CHECK (32_REG(1), OPERAND) ..CHECK FOR OPERAND BREAKPOINT.
19 RETURN
*****

```

OP_READ (REF 32_REG, M_DISPLACEMENT)

```

403 * 1  .. DETERMINE OPERAND ADDRESS FROM U REGISTER USING I, K, S, Y & B INDICATORS.
405 * 2  ..TRANSFER THE 32-BIT OPERAND (AS DETERMINED BY OPERAND TYPE) TO THE REGISTER SUPPLIED BY CALLER.
407 * 3  .. (1) 32_REG - 32-BIT REFERENCE REGISTER.
408 * 4  .. (2) M_DISPLACEMENT - 0/1 FOR OP1/OP2 SEQUENCE RESPECTIVELY.
409 * 5  IF U(1) "INDIRECT ADDRESSING", THEN
410 * 6  CALL IA_SEQUENCE(S_DESIGNATOR, Y, C, CL, P, MASK) ..FIND FINAL OPERAND ADDRESS VALUES.
412 * 7  OPERAND_DISPLACEMENT := OPERAND_DISPLACEMENT + M_DISPLACEMENT
413 * 8  ELSE ..COMPUTE FINAL OPERAND ADDRESS VALUES.
414 * 9  Y := U(Y)+B(U(8))(15) ..ADD DISPLACEMENT AND INDEX REGISTER.
415 * 10 Y := Y+M_DISPLACEMENT ..ADD DISPLACEMENT AND INDEX REGISTER.
417 * 11 S_DESIGNATOR := U(6)
418 * 12 IF CHARACTER_ADDRESSING_OVERRIDE, THEN
419 * 13 ACQUIRE ACTIVE C, P, AND MASK
420 * 14 ELSE
421 * 15 C := B(10) ..FORCE FOLLOWING CODE TO USE K DESIGNATOR.
422 * 16 EDDIE
423 * 17 EDDIE
424 * 18 IF .NOT. (INSTRUCTION_FORMAT_INDICATOR-I .AND. K=0 .AND. C EVEN) "NOT IMMEDIATE OPERAND", THEN
426 * 19 CALL SPR_CHECK (S_DESIGNATOR, OPERAND_READ, "P-1")
427 * 20 CALL ADD_S (S_DESIGNATOR, Y)
428 * 21 CALL BPF_CHECK (Y, OPERAND)
429 * 22 CALL MEMORY_READ (Y, 32_REG, S_DESIGNATOR, OPERAND, "P-1")
430 * 23 ELSE "HANDLE IMMEDIATE OPERAND"
431 * 24 32_REG := U(5)
432 * 25 IE C <> 0 .OR. CL <> 0, THEN ..IMMEDIATE OPERAND FOR INS INDIRECT ADDRESS DOESN'T USE
      32_REG := 32_REG + B(U(8))
434 * 26 EDDIE
435 * 27 EDDIE
436 * 28 IE C EVEN, THEN ..NOT CHARACTER ADDRESSING.
437 * 29 IF INSTRUCTION_FORMAT_INDICATOR-I, THEN
438 * 30 ADJUST 32_REG ACCORDING TO U(K) ..INCLUDES IMMEDIATE OPERANDS.
439 * 31 EDDIE
440 * 32 ELSE ..CHARACTER ADDRESSING.
441 * 33 RIGHT JUSTIFY / ZERO FILL 32_REG.
442 * 34 ..THIS CAN BE DONE BY SHIFTING 32_REG RIGHT LOGICAL P BITS & ANDING WITH MASK.
443 * 35 EDDIE
444 * 36 RETURN
445 * 37
446 *

```

OP_STORE (32_REG, M_DISPLACEMENT)

```

REF
PAGE *****
* 1 .. DETERMINE OPERAND ADDRESS FROM THE U REGISTER USING I, K, S, Y & B INDICATORS.
* 2 ..STORE THE QUANTITY SUPPLIED BY CALLER IN MEMORY.
* 3 .. (1) 32_REG - 32-BIT VALUE TO TRANSFER INTO MEMORY SUBJECT TO OPERAND TYPE RESTRICTIONS.
* 4 .. (2) M_DISPLACEMENT - 0/1 FOR OP1/OP2 SEQUENCE RESPECTIVELY.
* 5 IE U(1) "INDIRECT ADDRESSING", THEN
* 6 CALL 1A_SEQUENCE(S_DESIGNATOR, Y, C, C1, P, MASK) ..FIND FINAL OPERAND ADDRESS VALUES.
* 7 OPERAND_DISPLACEMENT := OPERAND_DISPLACEMENT + M_DISPLACEMENT
* 8 ELSE ..COMPUTE FINAL OPERAND ADDRESS VALUES.
* 9 Y := U(Y)+B(U(B))(15) ..ADD DISPLACEMENT AND INDEX REGISTER.
* 10 Y := Y+M_DISPLACEMENT ..ADD DISPLACEMENT FOR DESIRED OP1 OR OP2 SEQUENCE.
* 11 S_DESIGNATOR := U(S)
* 12 IE CHARACTER_ADDRESSING_OVERPIECE, THEN
* 13 ACQUIRE ACTIVE C, P AND MASK
* 14 ELSE
* 15 _C := 0'10' ..FORCE FOLLOWING CODE TO USE K DESIGNATOR.
* 16 ENDIE
* 17 ENDIE
* 18 SET MEMORY_STORE_INDICATOR ..FOR USE IN REPEAT TERMINATION DETECTION.
* 19 IE NOT WORD REFERENCE, THEN
* 20 IE .NOT.(INSTRUCTION_FORMAT_INDICATOR=I .AND. K=0 .AND. _C EVEN) "NOT IMMEDIATE OPERAND",
* 21 THEN
* 22 CALL SPR_CHECK (S_DESIGNATOR, OPERAND_WRITE, "P-1", Y)
* 23 CALL ADD_5 (S_DESIGNATOR, Y)
* 24 CALL DPR_CHECK (Y, OPERAND)
* 25 CALL MEMORY_READ (Y, VALUE, S_DESIGNATOR, OPERAND, "P-1")
* 26 IE _C EVEN, THEN ..NOT CHARACTER ADDRESSING
* 27 IE INSTRUCTION_FORMAT_INDICATOR = I, THEN
* 28 ADJUST 32_REG ACCORDING TO U(K) AND MASK INTO VALUE
* 29 ENDIE
* 30 ELSE ..CHARACTER ADDRESSING
* 31 ELSE POSITION CHARACTER AND MASK INTO VALUE
* 32 ENDIE
* 33 TRANSFER VALUE TO MEMORY(Y) ..STORE TO WORD LOCATION IS A MOOP.
* 34 ENDIE
* 35 RETURN
*
*****

```

NSWC

AN/UYK-7 (CP)

14 DEC 79 PAGE 16

* CP SUPPORT ROUTINES LEVEL II *

DPR_CHECK (ABS_ADDR, TYPE_)

REF
PAGE

```

    1  .. PERFORM BREAKPOINT REGISTER CHECKS. THE UYK7 EMULATION HAS EIGHT
    2  ..BREAKPOINT REGISTERS. ONE REGISTER CORRESPONDS TO THE ACTUAL HARDWARE
    3  ..BREAKPOINT REGISTER OF THE REAL UYK-7 MACHINE. THE OTHER SEVEN
    4  ..REGISTERS ARE PSEUDO BREAKPOINT REGISTERS AND ARE CONSIDERED
    5  ..EXTENSIONS OF THE UYK-7 ARCHITECTURE. THE PSEUDO BREAKPOINT REGISTERS
    6  ..ARE LOCATED IN CPP LOCATIONS 61-67 (PSEUDO BREAKPOINTS ARE NOT
    7  ..ACCESSIBLE TO UYK-7 PROGRAMS). IF PSEUDO BREAKPOINTS ARE ENABLED
    8  ..ALL BREAKPOINT REGISTERS WILL BE CHECKED (TOTAL OF 8), OTHERWISE, ONLY
    9  ..THE EMULATED HARDWARE REGISTER WILL BE CHECKED. TYPE_ WILL BE TESTED
    10  ..AGAINST THE TYPE FIELD OF THE ACTIVE BREAKPOINT REGISTERS AND IF THEY
    11  ..MATCH, ABS_ADDR WILL BE COMPARED TO THE BREAKPOINT REGISTER AND IF THEY
    12  ..ADDRESS BITS 0-171. FOR THE EMULATED HARDWARE REGISTER, A 'MIT' ON
    13  ..ADDRESS COMPARISON CAUSES A 'HALT' OR CLASS II INTERRUPT DEPENDING ON
    14  ..THE PROGRAM/MANUAL SWITCH. FOR THE PSEUDO BREAKPOINT REGISTERS, A
    15  ..'MIT' ON ADDRESS COMPARISON CAUSES A 'HALT' REGARDLESS OF THE
    16  ..PROGRAM/MANUAL SWITCH SETTING.
    17  ..(1) ABS_ADDR - 18 BIT ABSOLUTE UYK-7 ADDRESS
    18  ..(2) TYPE_ - TYPE OF BREAKPOINT CHECK DESIRED
    19  IE PSEUDO_BREAKPOINTS_ENABLED, THEN
    20  CNT := TOTAL # OF PSEUDO BREAKPOINTS ..CHECK ALL BREAKPOINT REGISTERS.
    21  ELSE
    22  CNT := 0 ..ONLY EMULATED HARDWARE BREAKPOINT CHECKED.
    23  ENDIE
    24  DO WHILE CNT >= 0 ..CYCLE THRU ACTIVE BREAKPOINT REGISTERS.
    25  IE CRR(BREAKPOINT_REGISTER+CNT)(19,23) <> 0, THEN
    26  IE (TYPE_ = CRR(BREAKPOINT_REGISTER+CNT)(19,23) OR
    27  CRR(BREAKPOINT_REGISTER+CNT)(19,23)=3) "MATCHING TYPES", THEN
    28  IE ABS_ADDR = CRR(BREAKPOINT_REGISTER+CNT)(17) "MATCHING ADDRESS", THEN
    29  IE MANUAL_MODE .OR. CNT > 0 "PSEUDO BPT.", THEN
    30  SET _STEP ..STOP IN MAIN LOOP IF OPERAND BREAKPOINT.
    31  IE TYPE_ = INSTRUCTION_BREAKPOINT, THEN
    32  INTERRUPT INSTRUCTION EXECUTION..STOP IMMEDIATELY IF INSTRUCTION TYPE-
    33  INTERRUPT INSTRUCTION EXECUTION.
    34  ELSE
    35  IE ASR(13,1) "CLASS II LOCKED OUT", THEN
    36  RETURN ..BREAKPOINT INTERRUPTS DISABLED.
    37  ELSE
    38  IE TYPE_ = INSTRUCTION_BREAKPOINT, THEN
    39  GENERATE SYNCHRONOUS_INTERRUPT ("P"-0, INSTRUCTION_BREAKPOINT)
    40  ..ABORT INSTRUCTION.
    41  ELSE
    42  GENERATE SYNCHRONOUS_INTERRUPT ("P"-1, OPERAND_BREAKPOINT)
    43  ..ABORT INSTRUCTION.
    44  ENDIE
    45  ENDIE
    46  ENDIE
    47  ENDIE
    48  ENDIE
    49  ENDIE
    50  CNT := CNT-1
    51  ENDDO ..END WHILE.
    52  RETURN

```

NSMC

IA_SEQUENCE (REF S_DESIGNATOR, REF Y, REF _C, REF _CL, REF _P, REF _MASK)

```

543 REF
544 PAGE
545
546 .. PERFORMS INDIRECT ADDRESSING RETURNING THE FOLLOWING VALUES:
547 .. (1) S_DESIGNATOR - S_DESIGNATOR OF OPERAND.
548 .. (2) Y - DISPLACEMENT PORTION OF OPERAND ADDRESS.
549 .. (3) _C, _P, _E, _MASK - CHARACTER ADDRESSING PARAMETERS WHEN NEEDED.
550 .. CALLER IS RESPONSIBLE FOR ENSURING THAT AN IA_SEQUENCE IS REQUIRED.
551
552 7 := L(Y) + B(L(Y))113)
553
554 S_DESIGNATOR := U(S)
555
556 DO WHILE U(1) = "INDIRECT" -- CASCADE IGNORING U(C).
557   CALL SPR_CHECK (S_DESIGNATOR, INDIRECT ADDRESSING, "P-1", Y)
558   32, REG(11) := Y -- SAVE Y FOR POSSIBLE COMPARISON TO SPR.
559   CALL ADD_S (S_DESIGNATOR, 32, REG(11))
560   CALL SPR_CHECK (32, REG(11), OPERAND)
561   CALL MEMORY_READ (32, REG(11), IACW, S_DESIGNATOR, OPERAND, "P-1") -- GET INDIRECT ADDRESS CONTROL
562   WORD (IACW).
563   U(19,20) := IACW(19,20) -- UPDATE THE U REGISTER BY I, S & Y FIELDS.
564   -- EXPERIMENTS INDICATE THAT U(19,17) IS ALWAYS SET TO 0 FOR THE LAST IACW IF THAT IACW IS OF
565   -- THE SPECIAL BASE (WS) TYPE. SINCE NO DOCUMENTATION CAN BE FOUND TO SUPPORT THIS
566   -- OBSERVATION IT IS OMITTED FROM THIS DESIGN.
567   DO CASE IACW(31,3) -- FORM NEW Y AND S_DESIGNATOR ACCORDING TO IACW.
568     .. 18*COO1A Y := U(S)
569     .. S_DESIGNATOR := U(8)
570     .. 13*OO1A Y := U(S) + 0(U(8))113)
571     .. S_DESIGNATOR := B(U(8))113)
572     .. ELSE Y := U(P) + B(U(8))113)
573     .. S_DESIGNATOR := U(S)
574   ENDCASE --END CASE
575   ENDDO --END WHILE
576   _C := IACW(C) -- RETURN DATA STRUCTURE INFORMATION TO CALLER.
577   _CL := IACW(CL) -- I.E. SAVE BIT 29 IN CASE =0, FMT I, E IMS OR IWB.
578   IE _C = 00 .AND. REPEAT_IN_PROGRESS --REPEAT OF SPECIAL INDIRECT, THEN
579     CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-2,CP ILLEGAL INSTRUCTION") --ABORT THE INSTRUCTION.
580   ENDO
581   IE _C = 00 .AND. U(P)=0*25* "SXB ILLEGAL IF I=1 AND C=00", THEN
582     CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-1,CP ILLEGAL INSTRUCTION") --ABORT THE INSTRUCTION.
583   ENDO
584   IE _C = 00 "CHARACTER REFERENCE", THEN
585     _P := IACW(P)
586     _MASK := IACW(M) 1 BITS RIGHT JUSTIFIED IN A 32-BIT WORD
587     SET CHARACTER_ADDRESSING_OVERWIDE
588     IE SEQUENTIAL CHARACTER ADDRESSING -AND- NOT WORD REFERENCE, THEN
589       -- MODIFY IACW(P) AND MAYBE IACW(Y).
590       IE (P-IACW(M)) < 0, THEN -- MUST UPDATE IACW(Y)
591         IACW(P) := 32 - IACW(M) IACW(Y) := IACW(Y) + 1
592       ELSE
593         IACW(P) := IACW(P) - IACW(M)
594       ENDO
595     ENDDO
596   ENDDO
597   ENDDO
598   RETURN
599

```

NSWC AN/UYM-7 (Co)
CP SUPPCH) ROUTINES LEVEL II

MEMORY_READ 18_REG, REF 32_REG, S_DESIGNATOR, RESUME_TYPE, P_MOD)

```

DEF
PAGE
*****
1  .. FETCH A 32-BIT QUANTITY FROM MEMORY(WORD OR MAIN) INTO THE REGISTER SUPPLIED BY THE CALLER.
2  .. (1) 18_REG - 18-BIT ABSOLUTE ADDRESS TO READ FROM.
3  .. (2) 32_REG - 32-BIT REFERENCE REGISTER.
4  .. (3) S_DESIGNATOR - BASE REGISTER ASSOCIATED WITH THE ADDRESS.
5  .. (4) RESUME_TYPE - DESIRED MEMORY RESUME ERROR IF ADDRESS IS OUT OF BOUNDS.
6  .. (5) P_MOD - P REGISTER MODIFICATION IF AN ERROR OCCURS.
7  .. (6) S_DESIGNATOR-7 .AND. ASR(7,1) .AND. 18_REG(512) "WORD REFERENCE", THEN
8  .. 32_REG := MCRG(18_REG)
9  ELSE ..FETCH FROM MAIN MEMORY.
10  .. IE 18_REG EXCEEDS MEMORY LIMIT, THEN
11  .. GENERATE SYNCHRONOUS INTERRUPT IP_MOD, RESUME_TYPE, 18_REG(16)
12  .. ..ABORT CURRENT INSTRUCTION.
13  ELSE
14  .. 32_REG := MAIN_MEMORY(18_REG)
15  ENDIE
16  ENDIE
17  RETURN
*****

```

SPR_CHECK (S_INDICATOR, CHECK_TYPE, P_MODIFICATION, I6_REG)

```

*****
REF  PAGE
618  * 1  .. IF PROTECTION CHECKING IS ENABLED (AS DETERMINED IN MAIN LOOP),
619  * 2  ..THIS ROUTINE PERFORMS THE ACCESS TYPE CHECK
620  * 3  ..DESIGNATED BY CALLER AND THE DISPLACEMENT CHECK. ON
621  * 4  ..ANY PROTECTION VIOLATION P IS MODIFIED AS INDICATED, THE
622  * 5  ..APPROPRIATE INTERRUPT GENERATED, AND THE INSTRUCTION
623  * 6  ..IS ABORTED. SOFTWARE PROTECTION REGISTER (SPR) FIELDS (E.G. IA, R,
624  * 7  ..IP, ETC.) ARE AS DEFINED ON THE REPERTOIRE CARD.
625  * 8  .. (1) S_INDICATOR - BASE REGISTER ASSOCIATED WITH THE ADDRESS
626  * 9  .. (2) CHECK_TYPE - SPECIFIES WHETHER OPERAND, INSTRUCTION OR INDIRECT ADDRESSING TYPE
627  * 10  .. (3) P_MODIFICATION - P REGISTER MODIFICATION IF AN ERROR OCCURS
628  * 11  .. (4) I6_REG - VALUE TO BE COMPARED AGAINST SPR LIMIT FIELD
629  * 12  IF SPR_CHECKS_ENABLED, THEN
630  * 13  FETCH APPROPRIATE SPR ..(I.E. CHR(160)+S_INDICATOR))
631  * 14  IF CHECK_TYPE VALID FOR THIS SPR, THEN
632  * 15  IF CHECK_TYPE = IA "INDIRECT ADDRESSING, THEN
633  * 16  IF SPR_PRIVILEGED_INSTRUCTION SET .AND. SPR(IP), THEN
634  * 17  ..SPECIAL PRIVILEGED INSTRUCTION(!! ON REP CARD) -> INST(I) SET AND SPR(I) SET
635  * 18  CALL GENERATE_SYNCHRONOUS_INTERRUPT(IP_MODIFICATION, PRIVILEGED_INSTRUCTION)
636  * 19  ..ABORT THE INSTRUCTION.
637  * 20  ELSE
638  * 21  IF ASR(11,2) .AND. SPR(IR) = 0, THEN
639  * 22  ..PROTECTION VIOLATION, ATTEMPT TO USE INTERRUPT B & S REGISTERS
640  * 23  ..FOR INDIRECT REFERENCE WITHOUT AUTHORIZATION.
641  * 24  CALL GENERATE_SYNCHRONOUS_INTERRUPT (P_MODIFICATION, IA OR OP READ)
642  * 25  ..ABORT THE INSTRUCTION.
643  * 26  ENDIE
644  * 27  ENDIE
645  * 28  IF I6_REG>SPR(R), THEN
646  * 29  GENERATE_SYNCHRONOUS_INTERRUPT ("P-" P_MODIFICATION, "APPROPRIATE LIMIT INTERRUPT")
647  * 30  ..ABORT CURRENT INSTRUCTION.
648  * 31  ENDIE
649  * 32  ELSE ..THERE IS A PROTECTION VIOLATION.
650  * 33  GENERATE_SYNCHRONOUS_INTERRUPT ("P-" P_MODIFICATION, "APPROPRIATE TYPE VIOLATION")
651  * 34  ..ABORT CURRENT INSTRUCTION.
652  * 35  ENDIE
653  * 36  ENDIE
654  * 37  RETURN
655  * 38
656  *
*****

```

NSWC

AN/UYK-7 (CP)

14 DEC 79 PAGE 21

* CP INSTRUCTION SUPPORT ROUTINES *
*

CP/IOC_CLOCK_COMMUNICATIONS

REF
PAGE

```
*****
1  .. THIS ROUTINE CAN BE USED BY THOSE CP INSTRUCTIONS REQUIRING A
2  .. RESPONSE (VIA THE O BUS) FROM THE IOC. AFTER INITIATING A REQUEST
3  .. TO THE IOC, THE FUNCTION DESIGNATORS ARE SENT TO THE IOC ACROSS THE
4  .. O BUS. THE IOC THEN RESPONDS WITH 32 BITS ON THE O BUS. THIS RESPONSE
5  .. WILL BE AVAILABLE TO THE CALLING INSTRUCTION.
6  .. INITIATE A REQUEST TO THE IOC DESIGNATED BY U(A)
7  .. ENSURE THAT THIS REQUEST HAS BEEN ACCEPTED.
8  O_BUS(31,6) := U(F) ..SEND FUNCTION DESIGNATORS FROM U.
9  O_BUS(22,3) := U(F2) ..TO INCLUDE SUBFUNCTION DESIGNATORS.
10 O_BUS(19,1) := 0 ..ENSURE THAT THIS REQUEST NOT INTERPRETED AS AN INTERRUPT STATUS CODE REQUEST.
11 SEND O_BUS TO THE IOC
12 WAIT UNTIL IOC HAS ACQUIRED THE INFORMATION FROM THE O_BUS
13 RECEIVE THE IOC'S RESPONSE ON THE O_BUS
14 RETURN ..INFORMATION NOW AVAILABLE ON O BUS.
*****
```

NSMC

AN/UYK-7 (CP)
CP INSTRUCTION SUPPORT ROUTINES

14 DEC 79 PAGE 23

DO_JUMP (S_DESIGNATOR, _DISPLACEMENT)

REF
PAGE

679
681
682
684
685
686
687

```

* 1  .. ROUTINE USED BY JUMP TYPE INSTRUCTIONS TO UPDATE THE PROGRAM COUNTER TO THE ADDRESS
* 2  --OF THE "JUMPED TO" INSTRUCTION. CALLED WHEN A JUMP IS TO BE "TAKEN".
* 3  .. (1) S_DESIGNATOR - BASE REGISTER INDICATOR FOR NEW PROGRAM COUNTER (P(S)).
* 4  .. (2) _DISPLACEMENT - DISPLACEMENT FOR NEW PROGRAM COUNTER (P(S)).
* 5  P(S) := S_DESIGNATOR
* 6  P(O) := _DISPLACEMENT
* 7  RETURN

```

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SUPPORT ROUTINES

GET_SHIFT_AMOUNT (M, REF_SHIFT_COUNT)

```

REF *****
PAGE *****
* 1 .. RETURN SHIFT COUNT INTERPRETED FROM N FIELD OF FMT IV B INSTRUCTION.
* 2 .. (1) M - M FIELD OF FMT IV B INSTRUCTION
* 3 .. (2) SHIFT_COUNT - SHIFT COUNT RETURNED TO CALLER
* 4 IE M(6:1) = 0, THEN
* 5   SHIFT_COUNT := M(5)
* 6 ELSE
* 7   IE M(5:1) = 0, THEN
* 8     CALL GET_BREG (M(3:3), 32_REG(1))
* 9     SHIFT_COUNT := 32_REG(1) * .6 BITS OF COUNT FOR DOUBLE SHIFTS.
* 10  ELSE
* 11    CALL GET_AREG (M(3:3), 32_REG(1))
* 12    SHIFT_COUNT := 32_REG(1) * .6 BITS OF COUNT FOR DOUBLE SHIFTS.
* 13  ENDIE
* 14  ENDIE
* 15  RETURN
* *****

```

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SUPPORT ROUTINES

HALF-WORD_TOGGLE

```

REF
PAGE
*****
* 1 .. UPDATE UPPER/LOWER DESIGNATOR (ASR(15,1)) UPON COMPLETION OF A HALF-WORD INSTRUCTION. CALLED
* 2 ..BY ALL HALF-WORD INSTRUCTIONS AFTER CHECKING FOR SYNCHRONOUS INTERRUPTS. THIS ALLOWS
* 3 ..THE RIGHT ASP TO BE STOPPED UPON INTERRUPT (ICP-34).
* 4 ..NOTE: A HALF-WORD INSTRUCTION EXECUTED REMOTELY DOES NOT AFFECT THE UPPER/LOWER DESIGNATOR.
* 5 IE .NOT. EXECUTE_REMOTE_IN_PROGRESS, THEN
* 6 IE ASR(15,1), THEN
* 7 IE ASR(15,1) := 0 ..NEXT INSTRUCTION FROM UPPER HALF.
* 8 ELSE
* 9 IE UL(F) >= 0'60' "ASSUME 2 HALF-WORD INSTRUCTIONS", THEN
* 10 ASR(15,1) := 1 ..EXECUTE LOWER HALF NEXT.
* 11 ENDIE
* 12 ENDIE
* 13 ENDIE
* 14 RETURN
*****

```

NSWC

AN/UVA-7 (CP)
CP INSTRUCTION SUPPORT ROUTINES

14 DEC 79 PAGE 26

REPLACE_CHECK (32_REG)

REF
PAGE

723
724
725
726
727
728
729

* 1 .. IF THE CURRENT INSTRUCTION IS A REPLACE TYPE, THEN UPDATE THE MEMORY
* 2 ..LOCATION (Y) SPECIFIED BY THE OPERAND ADDRESS.
* 3 .. (1) 32_REG - 32 BIT VALUE TO BE STORED IN MEMORY
* 4 IF UCF) >= 03, THEN ..THIS IS A REPLACE TYPE INSTRUCTION.
* 5 CALL _REPLACE (32_REG)
* 6 ENDIE
* 7 RETURN
*

MSUC

AN/UYK-7 (ICP)
CP INSTRUCTION SUPPORT ROUTINES

14 DEC 79 PAGE 27

-REPLACE (32_REG)

REF
PAGE

```

731 * 1 ** STORE 32_REG INTO THE MEMORY LOCATION (Y) SPECIFIED BY THE OPERAND
732 * 2 **ADDRESS. THE ALGORITHM USED TO COMPUTE THIS ADDRESS DIFFERS WHEN A
733 * 3 **REPEAT IS IN PROGRESS.
734 * 4 ** (1) 32_REG - 32 BIT VALUE TO BE STORED IN MEMORY
735 * 5 ** REPEAT_IN_PROGRESS AND. REPEAT_ADDR <> 0, THEN
736 * 6 ** SAVE UCS)
737 * 7 UCS) :- 6 **FORCE 56 ON STORE FOR REPEATED INSTRUCTIONS.
738 * 8 ** CALL OP_STORE (32_REG, "DISPLACE"0) **32_REG -> MEMORY
739 * 9 ** RESTORE UCS)
740 * 10 ELSE
741 * 11 ** CALL OP_STORE (32_REG, "DISPLACE"0) ** 32_REG -> MEMORY
742 * 12 ** ENDIE
743 * 13 ** RETURN

```

MSWC AN/UYK-7 (CP)
CP INSTRUCTION SUPPORT ROUTINES

UPDATEA_REPLACE (32_REG, A_DESIGNATOR)

745	REF	*****	
746	PAGE	*****	
747		* 1	.. UPDATE THE ACCUMULATOR A_DESIGNATOR AND PERFORM REPLACE INSTRUCTION
748		* 2	..CHECKS..
749		* 3	.. (1) 32_REG - 32 BIT REGISTER CONTAINING VALUE TO BE STORED
750		* 4	.. (2) A_DESIGNATOR - INTEGER NUMBER SPECIFYING THE ACCUMULATOR (A)
751		* 5	..REGISTER TO BE UPDATED
752		* 6	CALL PUT_AREG (A_DESIGNATOR, 32_REG)
		* 7	CALL REPLACE_CHECK (32_REG)
		* 8	RETURN
		*	*****

MSHC AN/UTK-7 (CP)
CP INSTRUCTION SUPPORT ROUTINES

SET_CD1 (QUANTITY1, QUANTITY2)

```

754 REF
755 PAGE
756
757
758
759
760
761
762
763
764
765
766
767
768
769
*****
1  .. COMPARE QUANTITY1 TO QUANTITY2 SETTING EQUAL/UNEQUAL AND GREATER OR EQUAL/LESS THAN
2  ..(ASR(2,2)) ACCORDINGLY.
3  .. (1) QUANTITY1 - 32-BIT VALUE.
4  .. (2) QUANTITY2 - 32-BIT VALUE.
5  IE QUANTITY1 = QUANTITY2, THEN
6  ASR(2,1) := 1 ..SET EQUAL INDICATION.
7  ELSE
8  ASR(2,1) := 0 ..SET NOT EQUAL INDICATION.
9  ENBIE
10 IE QUANTITY1 >= QUANTITY2. THEN
11 ASR(1,1) := 1 ..SET GREATER OR EQUAL INDICATION.
12 ELSE
13 ASR(1,1) := 0 ..SET LESS THAN INDICATION.
14 ENBIE
15 RETURN
*****

```

SET_C02 (QUANTITY1, QUANTITY2, QUANTITY3)

REF
PAGE771
773
774
775
776
777
778
779
780
781
782

```
*****
* 1  .. COMPARE QUANTITY1 TO QUANTITY2 AND QUANTITY3 SETTING OUTSIDE/WITHIN LIMITS
* 2  ..(ASR(0,1)) ACCORDINGLY.
* 3  .. (1) QUANTITY1 - 32-BIT REGISTER.
* 4  .. (2) QUANTITY2 - 32-BIT REGISTER.
* 5  .. (3) QUANTITY3 - 32-BIT REGISTER.
* 6  IF QUANTITY3 > QUANTITY1 .AND. QUANTITY1 >= QUANTITY2, THEN
* 7    ASR(0,1) = 0 ..SET WITHIN LIMITS INDICATION.
* 8  ELSE
* 9    ASR(0,1) = 1 ..SET OUTSIDE LIMITS INDICATION.
* 10  ENDOF
* 11  RETURN
*****
```

MSWC

AM/UYK-7 (CP)

14 DEC 79 PAGE 31

*
* CP UTILITY ROUTINES *
*

ADD_S (S_DESIGNATOR, REF Y)

REF
PAGE

```

785 * 1 .. ADD THE APPROPRIATE BASE(S) REGISTER TO THE DISPLACEMENT VALUE(Y).
786 * 2 .. (1) S_DESIGNATOR - INTEGER NUMBER SPECIFYING THE BASE(S) REGISTER
787 * 3 .. (2) SUPPLY INFORMATION (USED MODULO 8).
788 * 4 .. (3) Y - AN 18-BIT QUANTITY CONTAINING A ZERO EXTENDED 16-BIT VALUE (E.G. U(Y)+8(U(8)))
789 * 5 .. UPON ENTRY, THE 18-BIT RESULT OF THE ADDITION WILL BE RETURNED HERE.
790 * 6 .. NOTE: THE BASE REGISTER IS NOT ADDED ON WORD REFERENCES.
791 * 7 IE S_DESIGNATOR <> 7 -JP. ASP(721)=0 "NOT WORD REFERENCES", THEN
792 * 8 CALL GET_SREG (MODULO(S_DESIGNATOR*8), 32_REG(1)) ..GET DESIRED BASE REGISTER.
793 * 9 Y := Y + 32_REG(1) ..FORM 18-BIT VALUE.
794 * 10 ENDDIE
795 * 11 RETURN
796
797

```

MSWC

AN/UYK-7 (CP)
CP UTILITY ROUTINES

14 DEC 79 PAGE 33

GET_AREG (A_DESIGNATOR, REF 32_REG)

REF
PAGE

```

799 * 1 .. RETURN THE CONTENTS OF ACCUMULATOR A_DESIGNATOR IN 32_REG.
800 * 2 .. (1) A_DESIGNATOR - INTEGER NUMBER SPECIFYING THE ACCUMULATOR (A)
801 * 3 .. REGISTER TO SUPPLY INFORMATION (USED MODULO 8)
802 * 4 .. (2) 32_REG - 32 BIT REGISTER (SUPPLIED BY CALLER) RECEIVING
803 * 5 .. ACCUMULATOR VALUE
804 * 6 A_DESIGNATOR := MODULO(A_DESIGNATOR, 8)
805 * 7 IF ASP(10,1) "A78 REGISTER SELECT", THEN
806 * 8 32_REG := CHR(100+A_DESIGNATOR) .. INTERRUPT ACCUMULATOR SELECTED.
807 * 9 ELSE
808 * 10 32_REG := CHR(0+A_DESIGNATOR) .. TASK ACCUMULATOR SELECTED.
809 * 11 ENDIF
910 * 12 RETURN

```

GET_BREG (B_DESIGNATOR, REF 32_REG)

```

012 REF
014 PAGE
016 .....
017 * 1 .. LEAD THE LOW ORDER 19-BITS OF 32_REG WITH THE CONTENTS OF THE APPROPRIATE INDEX (B) REGISTER.
018 * 2 .. (1) B_DESIGNATOR - INTEGER NUMBER SPECIFYING THE INDEX (B) REGISTER.
019 * 3 .. TO SUPPLY INFORMATION (USED MODULO 8).
020 * 4 .. (2) 32_REG - 32-BIT REGISTER (SUPPLIED BY CALLER) RECEIVING RIGHT
021 * 5 .. JUSTIFIED 19-BIT QUANTITY FROM THE SPECIFIED INDEX REGISTER.
022 * 6 B_DESIGNATOR := MODULO(B_DESIGNATOR, 8)
023 * 7 IF B_DESIGNATOR = 0, THEN
024 * 8 32_REG(18) := 0 ..BIO) IS A SOURCE OF ZEROS.
025 * 9 ELSE
026 * 10 IF ASB(10,1) "A/B REGISTER SELECT", THEN
027 * 11 32_REG(18) := CBR(10*B_DESIGNATOR) ..I.E. APPROPRIATE INTERRUPT INDEX (B) REGISTER.
028 * 12 ELSE
029 * 13 32_REG(18) := CBR(10*B_DESIGNATOR) ..I.E. APPROPRIATE TASK INDEX (B) REGISTER.
030 * 14 ENDDIE
031 * 15 ENDDIE
032 * 16 RETURN
033 *
034 .....

```

GET_SREG IS_DESIGNATOR, REF 32_REG)

REF
PAGE

```

832 * 1 .. RETURN THE CONTENTS OF THE BASE REGISTER S_DESIGNATOR IN 32_REG.
833 * 2 ..(1) S_DESIGNATOR - BASE REGISTER DESIGNATOR
834 * 3 ..(2) 32_REG - 32 BIT REGISTER (SUPPLIED BY CALLER) RECEIVING
835 * 4 ..BASE REGISTER CONTENTS.
836 * 5 S_DESIGNATOR := MODULO(S_DESIGNATOR, 8)
837 * 6 IE ASR(1,1) "BASE(S) REGISTER SELECT", THEN
838 * 7 32_REG := CHR(120 + S_DESIGNATOR) ..INTERRUPT BASE REGISTER SELECTED.
839 * 8 ELSE
840 * 9 32_REG := CHR(20 + S_DESIGNATOR) ..TASK BASE REGISTER SELECTED.
841 * 10 ENDIE
843 * 11 RETURN
844 *

```

PUT_REG (A_DESIGNATOR, 32_REG)

REF
PAGE

```

046 * 1  .. STORE THE CONTENTS OF 32_REG INTO THE ACCUMULATOR A_DESIGNATOR.
047 * 2  .. (1) A_DESIGNATOR - INTEGER NUMBER SPECIFYING THE ACCUMULATOR (A)
048 * 3  ..REGISTER TO BE UPDATED (USED MODULO 8)
049 * 4  .. (2) 32_REG - 32 BIT VALUE TO BE STORED.
050 * 5  A_DESIGNATOR := MODULO(A_DESIGNATOR, 8)
051 * 6  IF ASR(10,1) "A/8 REGISTER SELECT", THEN
052 * 7  CHR(100+A_DESIGNATOR) := 32_REG ..INTERRUPT ACCUMULATOR SELECTED.
053 * 8  ELSE
054 * 9  CHR(10+A_DESIGNATOR) := 32_REG .. TASK ACCUMULATOR SELECTED.
055 * 10  ENDIF
056 * 11  RETURN

```

NSWC AN/UYK-7 (CP)
CP UTILITY ROUTINES

PUT_BREG (8_DESIGNATOR, 32_REG)

```

REF
PAGE *****
* 1  .. STORE THE LOWER 19 BITS OF 32_REG INTO THE APPROPRIATE INDEX REGISTER.
* 2  .. (1) 8_DESIGNATOR - INTEGER NUMBER SPECIFYING THE INDEX (8) REGISTER
* 3  ..TO BE UPDATED (USED MODULO 8).
* 4  .. (2) 32_REG - 32-BIT REGISTER CONTAINING RIGHT JUSTIFIED 19-BIT
* 5  ..QUANTITY TO BE STORED INTO THE SPECIFIED INDEX REGISTER.
* 6  8_DESIGNATOR := MODULO(8_DESIGNATOR, 8)
* 7  IF 8_DESIGNATOR <> 0, THEN --ATTEMPT TO STORE INTO B10) IS A MOOP.
* 8  IF ASR(10,1) "A/B REGISTER SELECT", THEN
* 9      CHR(110+8_DESIGNATOR) := 32_REG(10) --DEFINE INTERRUPT INDEX (8) REGISTER.
* 10  ELSE
* 11      CHR(110+8_DESIGNATOR) := 32_REG(10) --DEFINE TASK INDEX (8) REGISTER.
* 12  ENDIF
* 13  ENDIF
* 14  RETURN
*****

```

858
860
861
862
863
864
865
866
867
869
870
871
872
873

MSWC

AN/UYK-7 (CP)
CP UTILITY ROUTINES

14 DEC 79 PAGE 38

PUT_SREG (S_DESIGNATOR, 32_REG)

DEF

PAGE

075
076
077
078
079
080
081
082
083
084
085
086

```
* 1  ** STORE THE CONTENTS OF 32_REG INTO THE BASE REGISTER S_DESIGNATOR.
* 2  ** (1) S_DESIGNATOR - BASE REGISTER DESIGNATOR
* 3  ** (2) 32_REG - 32 BIT REGISTER CONTAINING VALUE TO BE STORED
* 4  S_DESIGNATOR := MODULO(S_DESIGNATOR, 8)
* 5  IF ASR(11,1) "BASE(S) REGISTER SELECT", THEN
* 6    CMR(120 + S_DESIGNATOR) := 32_REG -- INTERRUPT BASE REGISTER SELECTED.
* 7  ELSE
* 8    CMR(120 + S_DESIGNATOR) := 32_REG -- TASK BASE REGISTER SELECTED.
* 9  ENDIF
* 10 RETURN
*
```

NSWC

AM/UYK-7 (CP)

14 DEC 79 PAGE 39

```
*****  
*  
* FLOAING POINT SUBROUTINES *  
*  
*****
```

FLOATING_ADD_SUBTRACT_HEADER (DEF M_MANTISSA, REF A_MANTISSA)

```

REF PAGE *****
* 1 1 1 M_MANTISSA - 64 BIT REGISTER SUPPLIED BY CALLER TO CONTAIN MANTISSA FROM MEMORY.
* 2 2 1 A_MANTISSA - 64-BIT REGISTER SUPPLIED BY CALLER TO CONTAIN MANTISSA FROM A(U(A)+1).
* 3 3 1
* 4 4 1 SET SPR_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPR(16,1) SET AND U(1) SET.
* 5 5 1 CALL OP_READ (32_REG1), "DISPLACE=01" --GET CHARACTERISTIC FROM MEMORY.
* 6 6 1 CALL OP_READ (M_MANTISSA(63:32), "DISPLACE=11" --GET MANTISSA FROM MEMORY.
* 7 7 1 EXTEND SIGN BIT INTO M_MANTISSA(31:32)
* 8 8 1 CALL GET_AREG (U(A), 32_REG1) --GET CHARACTERISTIC FROM A(U(A)).
* 9 9 1 CALL GET_AREG (U(A)+1, A_MANTISSA(63:32)) --GET MANTISSA FROM A(U(A)+1).
* 10 10 1 EXTEND SIGN BIT INTO A_MANTISSA(31:32)
* 11 11 1 IF M_MANTISSA=ZERO "NEGATIVE OR POSITIVE", THEN
* 12 12 1 32_REG1 := 32_REG1 --ASSUME INTERMEDIATE CHARACTERISTIC THAT OF ACCUMULATOR VALUE.
* 13 13 1 ENDIF
* 14 14 1 IF A_MANTISSA=ZERO "NEGATIVE OR POSITIVE", THEN
* 15 15 1 32_REG1 := 32_REG1 --ASSUME INTERMEDIATE CHARACTERISTIC THAT OF MEMORY VALUE.
* 16 16 1 ENDIF
* 17 17 1 SHIFT_COUNT := 32_REG1-32_REG1 --ONES COMPLEMENT SUBTRACT.
* 18 18 1 IF SHIFT_COUNT > 0, THEN
* 19 19 1 SHIFT M_MANTISSA RIGHT ARITHMETIC SHIFT_COUNT PLACES
* 20 20 1 ELSE
* 21 21 1 IF SHIFT_COUNT < 0 "POSITIVE OR NEGATIVE", THEN
* 22 22 1 SHIFT_COUNT := .NOT. SHIFT_COUNT --ONES COMPLEMENT THE COUNT.
* 23 23 1 SHIFT A_MANTISSA RIGHT ARITHMETIC BY SHIFT_COUNT PLACES
* 24 24 1 ENDIF
* 25 25 1 CALL PUT_AREG (U(A), 32_REG1) --UPDATE INTERMEDIATE CHARACTERISTIC IN A(U(A)).
* 26 26 1 ENDIF
* 27 27 1 RETURN --INTERMEDIATE CHARACTERISTIC LEFT IN A(U(A)) FOR LATER USE!
*****

```

NSWC

AN/UYK-7 (CP)
FLOATING POINT SUBROUTINES

14 DEC 79 PAGE 41

FLOATING_OVERFLOW (REF 64_REG, REF 32_REG)

REF
PAGE

926
928
929
931
932
933
934

```
*****
* 1  ..(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER CONTAINING THE INTERMEDIATE RESULT OF A
* 2  ..FLOATING POINT OPERATION AFTER OVERFLOW HAS OCCURRED.
* 3  ..(2) 32_REG - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING THE INTERMEDIATE CHARACTERISTIC.
* 4  SHIFT 64_REG RIGHT 1 BIT --DIVIDE BY 2.
* 5  64_REG(63:1) := .NOT.64_REG(62:1) --RESTORE SIGN BIT.
* 6  32_REG := 32_REG +1 --INCREMENT THE CHARACTERISTIC.
* 7  RETURN
*****
```

FLOATING_NORMALIZE (REF 64_REG, REF 32_REG)

```

REF PAGE
*****
* 1 --(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER TO BE NORMALIZED AND STORED IN A(UXA)+1).
* 2 --(2) 32_REG - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING INTERMEDIATE CHARACTERISTIC.
* 3 IE 64_REG(63,1) "NEGATIVE", THEN
* 4 SHIFT_COUNT := (8 OF LEFT JUSTIFIED 1 BITS IN 64_REG) - 1
* 5 ELSE
* 6 SHIFT_COUNT := (8 LEFT JUSTIFIED 0 BITS IN 64_REG) - 1
* 7 ENDOF
* 8 IE SHIFT_COUNT < 0, THEN
* 9 SHIFT LEFT CIRCULARLY 64_REG BY SHIFT_COUNT
* 10 32_REG := 32_REG - SHIFT_COUNT --ADJUST CHARACTERISTIC (ONES COMPLEMENT)
* 11 ELSE
* 12 32_REG := 0
* 13 ENDOF
* 14 RETURN
*****

```

FLOATING_ROUND (PEF 64_REG, REF 32_REG)

REF
PAGE

```
*****
* 1 --(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER CONTAINING MANTISSA TO BE ROUNDED.
* 2 --(2) 32_REG - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING CHARACTERISTIC.
* 3 IE 64_REG(31,1)<>64_REG(63,1) "SIGN BIT", THEN
* 4 IE 64_REG(63,1) "NEGATIVE", THEN
* 5 64_REG(63,32) := 64_REG(63,32)-1
* 6 ELSE
* 7 64_REG(63,32) := 64_REG(63,32)+1
* 8 ENDIE
* 9 IE OVERFLOW OCCURRED, THEN
* 10 CALL FLOATING_OVERFLOW (64_REG, 32_REG)
* 11 ENDIE
* 12 ENCLIE
* 13 RETURN
*****
```

NSWC

AN/JUK-7 (CP)
FLOATING POINT SUBROUTINES

14 DEC 79 PAGE 44

ROUND_UP (REF MANTISSA, REF CHARACTERISTIC)

```

970 REF
972 PAGE
974
975
976
977
978
979
980
*****
* 1  ..(1) MANTISSA - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING MANTISSA TO BE ROUNDED UP.
* 2  ..(2) CHARACTERISTIC - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING CHARACTERISTIC.
* 3  MANTISSA := MANTISSA*1
* 4  IE MANTISSA(31:1) = 1 "OVERFLOW OCCURRED", THEN
* 5  ..NOTE THAT WORKING ONLY WITH POSITIVE QUANTITIES.
* 6  MANTISSA := MANTISSA.RL.1 ..ADJUST THE MANTISSA.
* 7  CHARACTERISTIC := CHARACTERISTIC+1 ..UPDATE THE CHARACTERISTIC.
* 8  ENDIE
* 9  RETURN
*****

```

MSWC

AN/UYK-7 (CPI)
FLOATING POINT SUBROUTINES

14 DEC 79 PAGE 45

DIVIDE_COMPARE (REF 64_REG, 32_REG)

REF
PAGE

```

982 * 1  --(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER CONTAINING DIVIDEND.
983 * 2  --(2) 32_REG - 32-BIT REGISTER CONTAINING DIVISOR.
984 * 3  IF 32_REG<=64_REG(63,32), THEN --USE 32-BIT UNSIGNED COMPARE.
985 * 4      64_REG(63,32) := 64_REG(63,32) - 32_REG --ONES COMPLEMENT SUBTRACT.
986 * 5      64_REG(6,1) := 1
987 * 6  ENDDIE
988 * 7  RETURN

```

MSHC

AN/UYK-7 (CP)
FLOATING POINT SUBROUTINES

14 DEC 79 PAGE 46

FLOATING_POINT_END (MANTISSA, CHARACTERISTIC)

REF
PAGE

```

990
991
992
993
994
995
996
997
998
999
1000
1001
1002

* 1 ** COMMON CODE FOR TERMINATION OF FLOATING POINT INSTRUCTIONS.
* 2 ** (1) MANTISSA - 32-BIT RESULTANT MANTISSA
* 3 ** (2) CHARACTERISTIC - 32-BIT RESULTANT CHARACTERISTIC.
* 4 ** IF MANTISSA=0 "RESULT ZERO", THEN
* 5 ** CHARACTERISTIC = 0 --ZERO THE CHARACTERISTIC.
* 6 ** ENDIE
36 * 7 CALL PUT_AREG (U(A)+1, MANTISSA)
36 * 8 CALL PUT_AREG (U(A), CHARACTERISTIC)
* 9 ** IF CHARACTERISTIC(15,1)<<CHARACTERISTIC(31,1)), THEN
11 * 10 CALL GENERATE_SYNCHRONOUS_INTERRUPT (=-P=-1, -FLOATING POINT OVERFLOW)
* 11 **ABORT THE INSTRUCTION..
* 12 ENDIE
* 13 RETURN
*
*****

```

MSMC

AM/UYK-7 (CP)

14 DEC 79 PAGE 47

* CP INSTRUCTION SET *

-OR(4 _ROR) ..(REPLACE) INCLUSIVE OR FMT 11 F-01 0 6 F-03 0

REF	PAGE
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

1101	2
1001	2
1007	3
1006	3
1001	4
5001	4

```

* CALL OP_READ(32_REGI1),"DISPLACE=0) --ACQUIRE OPERAND (Y) AND A(U(A)).
*
* 2 CALL GET_ARG(U(A)), REPEAT_ACCUMULATOR
*
* 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR -V. 32_REGI1 --PERFORM INCLUSIVE OR.
*
* 4 CALL UPDATE_A_REPLACE(REPEAT_ACCUMULATOR, U(A)) --RESULT GOES TO CMB & POSSIBLY TO MEMORY.
*
* 5 RETURN

```

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 49

--SCIE_RSC) --(REPLACE) SELECTIVE CLEAR & FMT II F=01 1 & F=03 1

```

REF
PAGE *****
14 * 1 CALL OP_READ (32_REG(1), "DISPLACE=0) --FETCH OPERAND (V) AND A(U(A)).
   * 2 32_REG(1) := --NOT. 32_REG(1) --COMPLEMENT Y.
   * 3 CALL GET_AREG(U(A), REPEAT_ACCUMULATOR)
   * 4 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR .A. 32_REG(1)
   * 5 CALL UPDATEA_REPLACE (REPEAT_ACCUMULATOR, U(A)) --RESULT GOES CMR & POSSIBLY TO MEMORY.
   * 6 RETURN
   *
*****

```

1014
1015
1016
1017
1018
1019
1020

MS(& RMS) ..(REPLACE) SELECTIVE SUBSTITUTE FMT II F=01 2 & F=03 2

```

*****
REF PAGE *****
*****
1022 14 * 1 CALL OP_READ(32_REG(1), "DISPLACE=0" --FETCH OPERAND (Y), A(U(A)), & A(U(A)+1).
1024 33 * 2 CALL GET_AREG(U(A), 32_REG(2))
1025 33 * 3 CALL GET_AREG(U(A)+2, REPEAT_ACCUMULATOR)
1026 * 4 32_REG(1) := 32_REG(1) -A. 32_PEG(2)
1027 * 5 REPEAT_ACCUMULATOR := (REPEAT_ACCUMULATOR -A. (WGT-32_REG(2))) -V. 32_REG(1)
1029 28 * 6 CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)+1) --RESULT GOES TO A(U(A)+1) & POSSIBLY TO MEMORY.
1031 * 7 RETURN
*****

```

NSWC

AN/UYK-7 (CPI)
CP INSTRUCTION SET

14 DEC 79 PAGE 51

_XOR(E_RXOP) --(REPLACE) EXCLUSIVE OR FMT II F-01 3 & F-03 3

```

REF
PAGE *****
1033 14 * 1 CALL OP_READ(32_REG(1),"DISPLACE=0") --FETCH OPERAND (Y) & A(U(A)).
1035 33 * 2 CALL GET_AREG(U(A), REPEAT_ACCUMULATOR)
1036 * 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR -X. 32_REG(1) --PERFORM EXCLUSIVE OR.
1038 28 * 4 CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)) --RESULT GOES TO A(U(A)) & POSSIBLY TO MEMORY.
1040 * 5 RETURN
*****

```

NSMC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 52

--ALP(6_RALP) --(REPLACE) ADD LOG. PROD. FMT II F-01 4 & F-03 4

REF

PAGE

1042
1044
1045
1046
1047
1048
1050

```
*****  
14 * 1 CALL OP_READ(REPEAT_ACCUMULATOR,"DISPLACE=0") --FETCH OPERAND (Y), A(U(A)) & A(U(A)+1).  
33 * 2 CALL GET_AREG(U(A), 32_REG(2))  
33 * 3 CALL GET_AREG(U(A)+1, 32_REG(3))  
* 4 REPEAT_ACCUMULATOR := (REPEAT_ACCUMULATOR .A. 32_REG(2)) + 32_REG(3)  
* 5 UPDATE_FIXED_POINT_OVERFLOW(ASR(3,1))  
28 * 6 CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)+1) --RESULT GOES TO CAR & POSSIBLY TO MEMORY.  
* 7 RETURN  
*****
```

MSWC

AN/UYK-7 (CPI)
CP INSTRUCTION SET

14 DEC 79 PAGE 53

_LLP (_MLP, _LLPM, _PLP, & _RMLP)

REF
PAGE

```

1052 * 1  .. LOAD, SUBTRACT, AND REPLACE LOGICAL PRODUCT.
1053 * 2  ..FMT II, F=01 5, 01 6, 01 7, 03 5, 03 6
1054 * 3  CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE=0")
1055 * 4  CALL GET_AREG (U(A), 32_REG(3))
1056 * 5  REPEAT_ACCUMULATOR := 32_REG(3) ..FORM LOGICAL PRODUCT.
1057 * 6  IE _MLP .OR. _RMLP, THEN
1058 * 7  CALL GET_AREG (U(A)+1, 32_REG(2))
1059 * 8  REPEAT_ACCUMULATOR := 32_REG(2) - REPEAT_ACCUMULATOR ..ONES COMPLEMENT SUBTRACT.
1060 * 9  UPDATE FIXED PCINT OVERFLOW BIT IN ASR(BIT 3)
1061 * 10 ENDIE
1062 * 11 IE _LLP, THEN
1063 * 12 CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR)
1064 * 13 ELSE
1065 * 14 CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)+1)
1066 * 15 ENDIE
1067 * 16 RETURN
1068 *
1069 *

```

MSMC

AN/JUK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 54

_CNT --COUNT ONES, FMT 11, F=02 0

REF
PAGE

```

1071 14 * 1
1072 * 2 CALL OP_READ (32_REG(2), "DISPLACE=0)
1073 * 3 REPEAT_ACCUMULATOR := 0
1074 * 4 I := 0
1075 * 5 DO _WHILE I < 32 --LOOP AND COUNT ONES.
1076 * 6 IF 32_REG(2){1,1}, THEN
1077 * 7 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 1
1078 * 8 ENDIE
1079 * 9 I := I + 1
1080 * 10 ENDDO
1081 * 11 CALL PUT_AREG (UYA), REPEAT_ACCUMULATOR) --STORE RESULTS
      * RETURN
      *

```

..XR (_XRL) ..EXECUTE REMOTE (LOWER), FMT II, F-02 2, F-02 3

REF

PAGE

```

1083 * 1 .._XR EXECUTES ONE INSTRUCTION ONLY (I, E, ONE FULL WORD INSTRUCTION
1084 * 2 ..OR UPPER HALF OF TWO HALF WORD INSTRUCTIONS).
1085 13 * 3 CALL JUMP_ADDRESS(=DISPLACE=0, OPERAND_S, OPERAND_DISPLACEMENT)
1086 * 4 ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS. OPERAND BASE REGISTER
1087 * 5 ..AND OPERAND DISPLACEMENT ARE RETURNED.
1088 * 6 32_REG(3) := OPERAND_DISPLACEMENT
1089 * 7 CALL ADD_S (OPERAND_S, 32_REG(3)) ..FIND REMOTE INSTRUCTIONS ABSOLUTE ADDRESS.
1090 * 8 CALL BPP_CHECK (32_REG(3), INSTRUCTION) ..PERFORM INSTRUCTION BREAKPOINT CHECKS.
1091 17 * 9 CALL SPR_CHECK (UCS), INSTRUCTION EXECUTE, "P-1", OPERAND_DISPLACEMENT)
1092 20 * 10 ..PERFORM INSTRUCTION SPR CHECKS
1093 * 11 OLOF2 := UC(2) ..SAVE _XR, _XRL INDICATOR.
1094 * 12 CALL MEMORY_READ (32_REG(3), U, UCS), INSTRUCTION, "P-1")
1095 * 13 ..FETCH REMOTE INSTRUCTION INTO U REGISTER.
1096 19 * 14 IE OLOF2 ODD, THEN .. ODD --> _XRL.
1097 * 15 UU := UL ..COPY LOWER HALF WORD INSTRUCTION TO UU.
1098 * 16 ENDIE
1099 * 17 SET EXECUTE_REMOTE_IN_PROGRESS
1100 * 18 DEFINE INSTRUCTION_FORMAT_INDICATOR ..I.E. I, II, III, OR IV.
1101 * 19 CALL _DECODE ..TRANSFER CONTROL TO APPROPRIATE INSTRUCTION.
1102 * 20 RETURN
1103
1104

```

MSMC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 56

_SLP ..STORE LOGICAL PRODUCT, FMT II, F-02 4

REF
PAGE

```

1106 33 * 1 CALL GET_AREG (U(A), REPEAT_ACCUMULATOR)
1107 33 * 2 CALL GET_AREG (U(A)+1, 32_REG(2))
1108 33 * 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR .A. 32_REG(2)
1109 15 * 4 CALL OP_STORE (REPEAT_ACCUMULATOR, "DISPLACE=0")
1110 15 * 5 RETURN

```

MSBC

ANALYZE-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 57

--SSUM --STORE SUM, FMT II, F-02 5

REF
PAGE

```
1112 33 * 1 CALL GET_AREG (U(A), BF, AT_ACCUMULATOR)
1113 33 * 2 CALL GET_AREG (U(A)+1, 32_REG(2))
1114 * 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(2)
1116 * 4 IF OVERFLOW OCCURRED, THEN
1117 * 5 ASR(3,2) := 1 --SET FIXED POINT OVERFLOW INDICATOR.
1118 * 6 ELSE
1119 * 7 ASR(3,1) := 0 --CLEAR FIXED POINT OVERFLOW INDICATOR.
1120 * 8 ENDIF
1121 36 * 9 CALL PUT_AREG (U(A)+1, REPEAT_ACCUMULATOR)
1122 15 * 10 CALL CP_STORE (REPEAT_ACCUMULATOR, "DISPLACE=0")
1123 * 11 RETURN
*****
```

-SELF .. STORE DIFFERENCE, FMT II, F=02 6

```

*****
REF PAGE *****
33 * 1 CALL GET_AREG(U{A},32,REG{1})
1126 33 * 2 CALL GET_AREG(U{A}+1,REPEAT_ACCUMULATOR)
1127 33 * 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32 * REG{1} --ONES COMPLEMENT.
1128 * 4 IF OVERFLOW OCCURRED, THEN
1129 * 5 ASR{3,1} := 1 --SET FIXED POINT OVERFLOW INDICATOR.
1130 * 6 ELSE
1131 * 7 ASR{3,1} := 0 --CLEAR FIXED POINT OVERFLOW INDICATOR.
1132 * 8 ENDIE
36 * 9 CALL PUT_AREG(U{A}+1,REPEAT_ACCUMULATOR)
1133 36 * 10 CALL OP_STORE(REPEAT_ACCUMULATOR,"DISPLACE=0")
1134 36 * 11 RETURN
1135 *
*****

```

NSWC AN/UYK-7 (CPI)
CP INSTRUCTION SET

_DS .. DOUBLE STORE ACCUMULATORS, FMT II, F=02 7

REF	PAGE	*****
1137	*	1 SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(1) SET.
1139	33 *	2 CALL GET_AREGU(A),32_REG(1))
1140	33 *	3 CALL GET_AREGU(A)+1,32_REG(2))
1141	15 *	4 CALL OP_STORE(32_REG(1), "DISPLACE=0)
1142	15 *	5 CALL OP_STORE(32_REG(2), "DISPLACE=1)
1143	*	6 RETURN

MSW

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 60

_ISF .. TEST 6 SET FLAG, FMT II, F=03 7

```

REF
PAGE .....
* 1 IE U(16,1) "INDIRECT ADDRESSING", THEN --NOT ALLOWED (AS PER REP CARD1).
* 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-1", CP_ILLEGAL_INST) --ABORT THE INSTRUCTION.
* 3 ENDIE
* 4 CALL OP_READ(REPEAT_ACCUMULATOR, "DISPLACE" 0)
* 5 IE REPEAT_ACCUMULATOR(31,1), THEN
* 6 ASR(2,1) := 0 --NOT EQUAL.
* 7 ELSE
* 8 ASR(2,1) := 1 --EQUAL.
* 9 ENDIE
* 10 REPEAT_ACCUMULATOR(31,1) := 1 --SET FLAG BIT.
* 11 CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE" 0)
* 12 RETURN
*
.....

```

MSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 61

-DL -- DOUBLE LOAD, FMT II F=05 0

REF
PAGE

```
1160 * 1 SET SPR_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPR(16,1) SET AND U(1) SET.
1161 * 2 CALL CP_FREG(32_REG(1), "DISPLACE=0)
1163 * 3 CALL CP_FREG(32_REG(2), "DISPLACE=1)
1164 * 4 PUT_AREG(1,32_REG(1))
1165 * 5 PUT_AREG(1,32_REG(2))
1166 * 6 RETURN
*****
```

MSWC

AM/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 62

DA (C DAN) .. DOUBLE ADD & SUBTRACT, FMT II, F-05 1 & F-05 2

REF
PAGE

```

1168 * 1
1170 * 2
1172 * 3
1173 * 4
1174 * 5
1175 * 6
1176 * 7
1177 * 8
1178 * 9
1179 * 10
1180 * 11
1181 * 12
1182 * 13
1183 * 14
1184 * 15
1185 * 16
1186 * 17
1187 * 18
1188 * 19

```

.. NOTE: 64_REG(1) AND 64_REG(2) ARE 64-BIT REGISTERS USED FOR DOUBLE LENGTH INSTRUCTION.
 CALL CP_READ(64_REG(2)(31,32), "DISPLACE=0")
 CALL CP_READ(64_REG(2)(63,32), "DISPLACE=1")
 GET_AREG(U(A), 64_REG(1)(31,32))
 GET_AREG(U(A)+1, 64_REG(1)(63,32))
 IF U(F2) = 1, THEN ..ADD.
 64_REG(1) := 64_REG(1) + 64_REG(2) ..ONES COMPLEMENT DOUBLE ADD.
 ELSE ..SUBTRACT.
 64_REG(1) := 64_REG(1) - 64_REG(2) ..ONES COMPLEMENT DOUBLE SUBTRACT.
 ENDIE
 IF OVERFLOW, THEN
 ASR(3,1) := 1 ..SET FIXED POINT OVERFLOW INDICATOR.
 ELSE
 ASR(3,1) := 0 ..CLEAR FIXED POINT OVERFLOW INDICATOR.
 ENDIE
 PUT_AREG(U(A), 64_REG(1)(31,32))
 PUT_AREG(U(A)+1, 64_REG(1)(63,32))
 RETURN

MSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 63

_DC -- DOUBLE COMPARE, FMT II, F= 05 3

```

1190 REF
1191 PAGE
1192
1193 * 1 -- NOTE: 64_REG(1) AND 64_REG(2) ARE 64-BIT REGISTERS USED FOR DOUBLE
1194 * 2 --LENGTH INSTRUCTIONS.
1195 * 3 SET SPP_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPP(16,1) SET AND UCI() SET.
1196 * 4 CALL GET_AREGU(A),64_REG(1)(63,32)
1197 * 5 CALL GET_AREGU(A),64_REG(1)(31,32)
1198 * 6 CALL OP_READ(64_REG(2)(63,32),"DISPLACE=1)
1199 * 7 CALL OP_READ(64_REG(2)(31,32),"DISPLACE=0)
1200 * 8 IF 64_REG(1) = 64_REG(2), THEN
1201 * 9 ASR(2,1) := 1 --SET EQUAL INDICATOR.
1202 * 10 ELSE
1203 * 11 ASR(2,1) := 0 --INDICATE NOT EQUAL.
1204 * 12 ENDIE
1205 * 13 IF 64_REG(1) >= 64_REG(2), THEN
1206 * 14 ASR(1,1) := 1 --INDICATE GREATER THAN OR EQUAL.
1207 * 15 ELSE
1208 * 16 ASR(1,1) := 0 --INDICATE LESS THAN.
1209 * 17 ENDIE
1210 * 18 RETURN
1211

```

--LMP --LOAD BASE & MEMORY PROTECTION, FMT II F-05 4

```

REF
PAGE *****
1210 14 * 1 CALL OP_READ (32_REG(1), "DISPLACE"), "DISPLACE"
1212 14 * 2 CALL OP_READ (32_REG(2), "DISPLACE")
1213 34 * 3 CALL GET_OREG (U(8), 32_REG(3))
1214 14 * 4 32_REG(3) := 32_REG(3) + U(Y)
1215 14 * 5 IE 32_REG(3) < 0 THEN "ODD ADDRESS", THEN
1216 14 * 6 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-1", CP_ILLEGAL_INSTRUCTION)
1217 14 * 7 --ABORT INSTRUCTION.
1218 14 * 8 ELSE
1219 14 * 9 IE ASR(19,4)=0 "TASK MODE" .AND. (.NOT. ASR(0,1) "NOT LOAD BASE ENABLE" .OR.
1221 14 * 10 U(5) < 7 .OR. U(3)=7), THEN
1222 14 * 11 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-1", PRIVILEGED_INSTRUCTION_VIOLATION)
1224 14 * 12 --ABORT INSTRUCTION.
1225 14 * 13 ENDIE
1226 14 * 14 CMR(0'20'+U(A)) := 32_REG(1)(17,18) --SET UP THE BASE REGISTER.
1227 14 * 15 CMR(0'160'+U(A)) := 32_REG(2)(20,21) --SET UP ASSOCIATED STORAGE PROTECTION REGISTER.
1229 14 * 16 CMR(0'170'+U(A))(19,3) := U(5) --SET UP BASE DESIGNATOR OF ASSOCIATED SIR.
1231 14 * 17 CMR(0'170'+U(A))(15,16) := 32_REG(3)(15,16) --SET UP DISPLACEMENT OF ASSOCIATED SIR.
1233 14 * 18 ENDIE
1234 14 * 19 RETURN
*****

```

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_FA (_FAR) ..FLOATING-POINT ADD (WITH ROUND), FMT II, F=06 0 5 06 4

REF	PAGE	
1236	*	CALL FLOATING_ADD_SUB_MODE (64_REG(1), 64_REG(2))
1237	*	64_REG(2) := 64_REG(2) + 64_REG(1) ..64-BIT ONE'S COMPLEMENT ADD.
1238	33	CALL GET_AREG (U(A), 32_REG(1)) ..GET INTERMEDIATE CHARACTERISTIC.
1239	*	IF OVERFLOW OCCURRED, THEN
1240	41	CALL FLOATING_OVERFLOW (64_REG(2), 32_REG(1))
1241	*	ELSE
1242	42	CALL FLOATING_NORMALIZE (64_REG(2), 32_REG(1))
1243	*	ENDIF
1244	*	IF U(F2)=4 "ROUNDING OPTION", THEN
1245	43	CALL FLOATING_ROUND (64_REG(2), 32_REG(1))
1246	*	ENDIF
1247	46	CALL FLOATING_POINT_END (64_REG(2)(63:32), 32_REG(1))
1248	*	RETURN

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 66

_FAR (_FAHR) --FLOATING-POINT SUBTRACT (ROUND), FMT II, F-C6 1 & 06 5

REF
PAGE

1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262

```

* 1 CALL FLOATING_ADD_SUB_MDR (64_REG(1), 64_REG(2))
* 2 64_REG(2) := 64_REG(2) - 64_REG(1) --64-BIT ONE'S COMPLEMENT SUBTRACT.
33 * 3 CALL GET_AREG (U(3), 32_REG(1)) --GET INTERMEDIATE CHARACTERISTIC.
* 4 IF OVERFLOW OCCURRED, THEN
41 * 5 CALL FLOATING_OVERFLOW (64_REG(2), 32_REG(1))
1255 * 6 ELSE
42 * 7 CALL FLOATING_NORMALIZE (64_REG(2), 32_REG(1))
1257 * 8 ENDIE
* 9 IF U(2) = F "ROUNDING OPTION", THEN
43 * 10 CALL FLOATING_ROUND (64_REG(2), 32_REG(1))
1259 * 11 ENDIE
46 * 12 CALL FLOATING_POINT_END (64_REG(2)(63,32), 32_REG(1))
* 13 RETURN
*

```

MSVC AN/UYK-7 (CP)
CP INSTRUCTION SET

_FN (_FNR) ..FLOATING POINT MULTIPLY (ROUND), FMT II, F-06 2 & 06 6

```

1264 REF *****
1265 PAGE *****
1266
1267 * 1 SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND UC(1) SET.
1268 * 2 CALL OP_READ (32_REG(4), "DISPLACE=0") ..GET MULTIPLIER CHARACTERISTIC FROM MEMORY.
1269 * 3 CALL OP_READ (32_REG(1), "DISPLACE=1") ..GET MULTIPLIER MANTISSA FROM MEMORY.
1270 * 4 CALL GET_AREG (UC(2), 32_REG(3)) ..GET MULTIPLICAND CHARACTERISTIC FROM AUC(2).
1271 * 5 CALL GET_AREG (UC(2)+1, 32_REG(2)) ..GET MULTIPLICAND MANTISSA FROM AUC(2)+1.
1272 * 6 SIGN_INDICATOR := 32_REG(1)(31,1) XOR. 32_REG(2)(31,1) ..SAVE SIGN FOR RESULT.
1273 * 7 FORCE 32_REG(1) "MULTIPLIER" POSITIVE
1274 * 8 IF 32_REG(1) = 0 , THEN ..MULTIPLY BY ZERO.
1275 * 9 CALL PUT_AREG (UC(2), 32_REG(1))
1276 * 10 CALL PUT_AREG (UC(2)+1, 32_REG(1))
1277 * 11 RETURN
1278 * 12
1279 * 13 32_REG(3) := 32_REG(3) + 32_REG(4) ..FORM THE NEW CHARACTERISTIC.
1280 * 14 CALL DO_MULTIPLY ..32_REG(2) X 32_REG(1) WITH 64-BIT RESULT PUT IN REPEAT_ACCUMULATOR &
1281 * 15 32_REG(1).
1282 * 16 SHIFT REPEAT_ACCUMULATOR AND 32_REG(1) PAIR LEFT LOGICAL ONE ..CORRECT FOR FRACTIONAL MULTIPLY.
1283 * 17 IE REPEAT_ACCUMULATOR(30,1) = 0 , THEN ..NORMALIZE.
1284 * 18 SHIFT REPEAT_ACCUMULATOR AND 32_REG(1) PAIR LEFT LOGICAL ONE
1285 * 19 32_REG(3) := 32_REG(3) - 1 ..UPDATE CHARACTERISTIC.
1286 * 20
1287 * 21 IE U(F2) = 6 "ROUNDING OPTION", THEN
1288 * 22 CALL FLOATING_ROUND (REPEAT_ACCUMULATOR, 32_REG(3))
1289 * 23
1290 * 24 IE SIGN_INDICATOR "NEGATIVE", THEN ..DO SIGN CORRECTION.
1291 * 25 REPEAT_ACCUMULATOR := .NOT. REPEAT_ACCUMULATOR
1292 * 26
1293 * 27 CALL FLOATING_POINT_END (REPEAT_ACCUMULATOR, 32_REG(3))
1294 * 28 RETURN
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```

_F01_F08) --FLOATING-POINT DIVIDE (WITH ROUND), FMT II, F=06 3 & 06 7

```

1300
1302 * 1 SET SPR_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPR(10,1) SET AND UC(1) SET.
1304 * 2 CALL OP_READ (32_REG(2), "DISPLACE-0") --GET DIVISOR CHARACTERISTIC FROM MEMORY.
1306 * 3 CALL GET_AREG (U(3), 32_REG(3)) --GET DIVISOR NANTISSA FROM MEMORY.
1308 * 4 CALL GET_AREG (U(3), 64_REG(3)) --GET DIVIDEND CHARACTERISTIC FROM AUC(3).
1310 * 5 CALL GET_AREG (U(3)+1, 64_REG(1)(63,32)) -- GET DIVIDEND NANTISSA FROM AUC(3)+1).
1312 * 6 SIGN_IND := 64_REG(1)(63,1) XOR 32_REG(1)(31,1) --SIGN INDICATOR.
1314 * 7 FORCE 64_REG(1) "DIVIDEND" AND 32_REG(1) "DIVISOR" POSITIVE
1316 * 8 IE 64_REG(1)(63,32) "DIVIDEND" = 0, THEN
1318 * 9 32_REG(3) := 0 --FORCE DIVIDEND CHARACTERISTIC TO ZERO.
1320 * 10 ENBIE
1322 * 11 IE 32_REG(1) = 0 "DIVISOR ZERO", THEN
1324 * 12 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-1, FLOATING POINT OVERFLOW)
1326 * 13 ENBIE
1328 * 14 32_REG(3) := 32_REG(3) - 32_REG(2) --COMPUTE INTERMEDIATE CHARACTERISTIC.
1330 * 15 CALL DIVIDE_COMPARE (64_REG(1), 32_REG(1))
1332 * 16 COUNT := 31
1334 * 17 DO WHILE COUNT > 0
1336 * 18 64_REG(1) := 64_REG(1) .LL. 1
1338 * 19 CALL DIVIDE_COMPARE (64_REG(1), 32_REG(1))
1340 * 20 COUNT := COUNT-1
1342 * 21 ENDDO --END WHILE.
1344 * 22 64_REG 1= 64_REG SHIFTED LEFT CIRCULAR 32 BITS
1346 * 23 --64_REG CAN NOW SHARE TERMINATION LOGIC WITH OTHER FLOATING POINT ROUTINES.
1348 * 24 --NOTE, 64_REG(63,32) NOW HAS THE QUOTIENT AND 64_REG(31,32) HAS THE REMAINDER.
1350 * 25 IE U(2)-3 "NO ROUNDING", THEN
1352 * 26 IE 64_REG(1)(63,1)-1 "OVERFLOW OCCURRED", THEN
1354 * 27 CALL FLOATING_OVERFLOW (64_REG(1), 32_REG(3)) --UPDATE NANTISSA AND CHARACTERISTIC.
1356 * 28 ENBIE
1358 * 29 ELSE --"ROUNDING".
1360 * 30 IE 64_REG(1)(63,1)-1 "OVERFLOW OCCURRED", THEN
1362 * 31 ROUND_BIT := 64_REG(1)(32,1) --SAVE ROUND INDICATOR.
1364 * 32 CALL FLOATING_OVERFLOW (64_REG(1), 32_REG(3)) --UPDATE NANTISSA AND CHARACTERISTIC.
1366 * 33 IE ROUND_BIT SET, THEN --ROUND UP.
1368 * 34 CALL ROUND_UP (64_REG(1)(63,32), 32_REG(3))
1370 * 35 ENBIE
1372 * 36 ELSE --NO OVERFLOW.
1374 * 37 64_REG(1)(31,32) := 64_REG(1)(31,32) - 132_REG(11/2) --REMAINDER MINUS 1/2 THE DIVISOR.
1376 * 38 IE 64_REG(1)(31,32) POSITIVE, THEN --ROUND UP.
1378 * 39 CALL ROUND_UP (64_REG(1)(63,22), 32_REG(3))
1380 * 40 ENBIE
1382 * 41 ENBIE
1384 * 42 ENBIE
1386 * 43 IE SIGN_IND SET, THEN --CORRECT SIGN.
1388 * 44 64_REG(1)(63,32) := .NOT. 64_REG(1)(63,32)
1390 * 45 ENBIE
1392 * 46 CALL FLOATINT_POINT_END (64_REG(1)(63,32), 32_REG(3)) --FINISH THE INSTRUCTION.
1394 * 47 RETURN

```

NSWC AN/UYK-7 (CPI)
CP INSTRUCTION SET

_XS ..ENTER EXECUTIVE STATE, FMT II F-07 0 A-0

REF	PAGE	
1360		
1361	18	1 IE U(I) "INDIRECTION", THEN
1363	34	2 CALL IA_SEQUENCE(DUMMY PARAMETERS) --OP_READ NOT USED SINCE OPERAND IS NOT FETCHED.
1365	11	3 ENDIE
1367		4 CALL GET_BREG (U(8),32_REG(1))
1369		5 CODE := U(SY) + 32_REG(1)(15,16) --16-BIT ISC TG BE STORED UPON INTERRUPT.
		6 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-0, ENTER EXECUTIVE STATE, CODE)
		7 RETIE
		

__IPI __INTERPROCESSOR INTERRUPT, FMT II F-07 0 A-1

```

REF
PAGE
*
* 1  IF ASR(19,4)=0 "TASK MODE", THEN
1371
1372 11 * 2  CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-1", PRIVILEGED INSTRUCTION VIOLATION)
1373
1374 * 3  __ABORT INSTRUCTION.
1375
1376 * 4  ELSE
1377
1378 * 5  IF UC(1) = "IMDICTION", THEN
1379
1380 * 6  CALL IA_SEQUENCE (DUMMY PARAMETERS) __BP_READ NOT USED SINCE OPERAND IS NOT FETCHED.
1381
1382 * 7  ENDIE
1383
1384 * 8  CALL GET_OREG (UC(5), 32_REG(1))
1385
1386 * 9  32_REG(1) := UC(5) + 32_REG(1) __16-BIT QUANTITY.
1387
1388 * 10  IF 32_REG(1)(15,1), THEN
1389
1390 * 11  32_REG(1)(CP,1) := 0 __PREVENT INTERRUPTION OF SELF.
1391
1392 * 12  ENDIE
1393
1394 * 13  I := 0
1395
1396 * 14  DO WHILE I<8
1397
1398 * 15  IF 32_REG(1)(1,1), THEN
1399
1400 * 16  SEND INTERPROCESSOR INTERRUPT TO CP(1)
1401
1402 * 17  ENDIE
1403
1404 * 18  I := I+1
1405
1406 * 19  ENDDO __END WHILE.
1407
1408 * 20  ENDIE
1409
1410 * 21  RETURN
1411
*

```

MSMC AN/UYK-7 (CP)
CP INSTRUCTION SET

_AEI (E _PEI) ..ALLOW & PREVENT ENABLE INTERRUPTS, FMT II F-07 1 & 2

```

1395 * 1 IE ASRC(9,4)=0 "TASK MODE", THEN
1396 * 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P=-1, PRIVILEGED INSTRUCTION VIOLATION")
1397 * 3 ..ABORT INSTRUCTION.
1398 * 4 ELSE
1399 * 5 IE UC(1) "INDIRECTION", THEN
1400 * 6 CALL IA_SEQUENCE (DUMMY PARAMETERS) ..OP_READ NOT USED SINCE OPERAND IS NOT FETCHED.
1401 * 7 ENDIE
1402 * 8 CALL GET_BREG(UC(8), 32_REG(1))
1403 * 9 32_REG(1) := UC(5) + 32_REG(1) ..COMPUTE LOW 16-BITS OF O_BUS.
1404 * 10 INITIATE AN IOC REQUEST ON IOC(UC(4).A.3)
1405 * 11 O_BUS := (UC(7).LL.26) + (UC(2).LL.20) + 32_REG(1)(15,16) ..FUNCTION CCE AND VALUE TO O_BUS.
1406 * 12 SEND O_BUS TO IOC(UC(4).A.3)
1407 * 13 IE REQUEST NOT HONORED ..OR, O_BUS NOT RECEIVED WITHIN ACCEPTABLE TIME FRAME, THEN
1408 * 14 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P=-1, CP-I/O COMMAND RESUME, UC(3))
1409 * 15 ..ABORT INSTRUCTION.
1410 * 16 ENDIE
1411 * 17 ENDIE
1412 * 18 RETURN
1413 *
1414 *
1415 *
1416 *

```

AN/UYK-7 (CP)
CP INSTRUCTION SET

NSWC

_LIM --LJAD, ENABLE IOC MONITOR CLOCK, FMT II F-07 3

```

REF
PAGE
*****
* 1 IE ASFC(10,4)=0 "TASK MODE", THEN
* 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P=-1, PRIVILEGED INSTRUCTION VIOLATION)
* 3 --ABORT THE INSTRUCTION.
* 4 ELSE
* 5 IE U(1) "INDIRECTION", THEN
* 6 CALL IA_SEQUENCE (DUMMY PARAMETERS) --OP_PEAID NOT USED SINCE OPERAND IS NOT FETCHED.
* 7 ENDIE
* 8 CALL GET_BREGU(0), 32_REG(11)
* 9 32_REG(1) := U(SY) + 32_REG(1) --COMPUTE LGW 16-BITS OF O_BUS.
* 10 INITIATE AN IOC REQUEST ON IOC(U(A),A,3)
* 11 O_BUS := (U(F).LL.26) + (U(F2).LL.20) + 32_REG(1)(15,16) --FUNCTION CODE AND VALUE TO O_BUS.
* 12 SEND O_BUS TO IOC(U(A),A,3)
* 13 IE REQUEST NOT HONORED -OR- O_BUS NOT RECEIVED WITHIN ACCEPTABLE TIME FRAME, THEN
* 14 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P=-1, CP-IOC COMMAND RESUME, U(A))
* 15 --ABORT INSTRUCTION.
* 16 ENDIE
* 17 ENDIE
* 18 RETURN
*
*****
1410
1419
1421
1422
1423
1424
1426
1427
1428
1429
1430
1432
1433
1435
1436
1437
1438
1439

```

_IO ..INITIATE I/O. FMT II F-07 4

```

REF
PAGE
*****
1441 * 1 IE ASFC(16,4)=0 "TASK MODE", THEN
1442 * 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P=-1, PRIVILEGED INSTRUCTION VIOLATION)
1443 * 3 ..ABORT THE INSTRUCTION.
1444 * 4
1445 * 5 ELSE
1446 * 6 IE U(1) "INDIRECT ADDRESSING", THEN
1447 * 7 CALL IA_SEQUENCE (S,DESIGNATOR, Y, DUMMY PARAMETERS) ..DO CASCAADING AS REQUIRED.
1448 * 8 ELSE ..DEFINE NORMAL BASE (S) REGISTER DESIGNATOR & DISPLACEMENT.
1449 * 9 S_DESIGNATOR := U(5) ..DEFINE BASE (S) REGISTER SELECTOR.
1450 * 10 CALL GET_RREG (U(6), 32,REG11)
1451 * 11 Y := U(1) + 32_REG(11)(15) ..COMPUTE DISPLACEMENT.
1452 * 12
1453 * 13 ENDDIE
1454 * 14 CALL ADD_S (S,DESIGNATOR, Y) ..COMPUTE ABSOLUTE ADDRESS.
1455 * 15 INITIATE_AN_I/O_REQUEST_ON_IOC(U(A).A.3)
1456 * 16 C_BUS := U(F).LL.26 + U(F2).LL.20 + Y(17) ..FUNCTION DESIGNATORS & ABSOLUTE ADDRESS.
1457 * 17 SEND_C_BUS_TO_IOC(U(A).A.3)
1458 * 18 IE .NET.(I/O REQUEST MONITOR) ..OR. (I/O REQUEST) WITHIN ACCEPTABLE TIME FRAME, THEN
1459 * 19 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P=-1, IOC COMMAND RESUME, U(A).A.3)
1460 * 20 ..ABORT THIS INSTRUCTION.
1461 * 21 ENDDIE
1462 * 22 ENDDIE
1463 * 23 RETURN
1464 * 24
1465 * 25
1466 * 26
*****

```

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 74

_IR --INTERUPT RETURN, FMT II F-07 5

1468	REF	*****	* 1	IF ASP(19,4)=0 "TASK MODE", THEN	*****
1469	PAGE	*****	* 2	CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P--1, PRIVILEGED INSTRUCTION VIOLATION)	*****
1471		*****	* 3	--ABORT THIS INSTRUCTION.	*****
1472		*****	* 4	ELSE	*****
1473		*****	* 5	_CLASS := CURRENT_INTERRUPT_LEVEL (STATE) --AS INDICATED BY ACTIVE STATUS REGISTER.	*****
1475		*****	* 6	ASR := CMB(0'135'*(4*_CLASS) --PESTORE ACTIVE STATUS REGISTER.	*****
1476		*****	* 7	P := CMB(0'137'*(4*_CLASS) --RESTORE PROGRAM COUNTER.	*****
1477		*****	* 8	ENDIE	*****
1478		*****	* 9	RETURN	*****
		*****	*		*****

_RP ..REPEAT, FMT II F-07 6

REF
PAGE1480
1481
1482
1483
1484
1485
1486
1487
1488
1489

1491
1493
1494
1495
1496
1497

```
*****
* 1 .. THE INSTRUCTION SEQUENCE WILL CHECK FOR REPEATABILITY OF THE NEXT
* 2 ..INSTRUCTION, AND IF REPEATABLE THE REPEAT INDICATOR WILL BE SET.
* 3 ..SHOULD THE NEXT INSTRUCTION BE FOUND NOT TO BE REPEATABLE (AS PER
* 4 ..THE REPETITIOIRE CARD), THAT INSTRUCTION WILL BE ABOPTED AND REPEAT
* 5 ..MODE TERMINATED.
* 6 IE U(1) "INDIRECT ADDRESSING", THEN
* 7 CALL IA_SEQUENCE (DUMPY PAPAMETERS) ..DO CASCAADING AS REQUIRED.
* 8 ENDIE
* 9 IE B(7) = 0, THEN ..SKIP NEXT INSTRUCTION.
* 10 P(0) := P(0) + 1 ..INCREMENT PROGRAM COUNTER TO THE ADOESS JUST AFTER THAT OF THE REPEATED
* .. INSTRUCTION.
* 11 ELSE ..EXECUTE NEXT INSTRUCTION B(7) TIMES OR UNTIL THE CONDITION INDICATED BY U(A).
* 12 SET REPEAT_PENDING INDICATOR
* 13 SAVE U(A) & U(8) IN REPEAT_AB FOR REPEAT SEQUENCE USAGE
* 14 SAVE U(SY) IN REPEAT_SY
* 15 ENDIE
* 16 RETURN
*****
```

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 76

--LA --LOAD ACCUMULATOR WITH MEMORY CONTENTS, FMT I F-10

REF
PAGE

1499
1501
1502

14 * 1
36 * 2
* 3
*

CALL CP_READ (REPEAT_ACCUMULATOR, "DISPLACE=0") --FETCH OPERAND AS PER K DESIGNATOR.
CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR)
BEIJEN

LX8 ..LOAD ACCUMULATOR E INDEX B, FMT 1, F=11

```

*****
REF  *****
PAGE *****
*****
1504 * 1 SET SPR_PRIVILEGED_INSTRUCTION_INDICATOR ..LBO IS A SPECIAL CASE FOR SPR(10,1)-1 & INDIRECT
      * ADDRESSING.
      *
1506 * 2 CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE=01)
1507 * 3 CALL PUT_ARG (U(A), REPEAT_ACCUMULATOR)
1508 * 4 CALL GET_ARG (U(E), 32_REG(1))
1509 * 5 32_REG(1) := 32_REG(1)IS) + 1 ..INCREMENT INDEX (B) REGISTER VALUE.
1510 * 6 CALL PUT_ARG (U(B), 32_REG(1))
1511 * 7 RETURN
      *
*****

```

MSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 78

_LOIF --LOAD DIFFERENCE, FMT I, F-12

REF
PAGE

1513 14 *
1514 33 *
1515 *
1517 *
1518 36 *
1519 *
1519 *

1 CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE= 0)
2 CALL GET_AREG (UCA), 32-REG(2))
3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR - 32-REG(2) --ONES COMPLEMENT SUBTRACT.
4 UPDATE_FIXED_PCINT_OVERFLOW BIT IN ASR (BIT 3)
5 CALL PUT_AREG (UCA)+1, REPEAT_ACCUMULATOR
6 RETURN

NSWC

AN/LYN-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 79

_ANA --SUBTRACT A, FMT I, F-13

DEF
PAGE

1521
1522
1523
1525
1526
1527

14 * 1 CALL OP_READ (32_REG, "DISPLACE=0")
33 * 2 CALL GET_AREG (U(A), REPEAT_ACCUMULATOR)
* 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR - 32_REG(2) --ONES COMPLEMENT SUBTRACT.
* 4 UPDATE_FIXED_POINT_OVERFLOW_BIT IN ASR (BIT 3)
36 * 5 CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR)
* 6 RETURN
*

NSWC AN/UYK-7 (CPI)
CP INSTRUCTION SET

_AA --ADD A, FMT I, F-14

1529	REF	*****	
1530	PAGE	*****	
1531		*****	
1533		*****	
1534		*****	
1535		*****	

14 *	1	CALL OP_READ (32_REG(2), "DISPLACE" 0)
33 *	2	CALL GET_AREG (UCA), REPEAT_ACCUMULATOR
	3	REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(2) --ONES COMPLEMENT ADDITION.
	4	UPDATE_FIXED_POINT_OVERFLOW_BIT_IN_ASR (BIT 3)
36 *	5	CALL PUT_AREG (UCA), REPEAT_ACCUMULATOR
	6	RETURN

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_LSUM ..LOAD SUM, FMT I, F-15

```

REF
PAGE *****
1537 14 * 1 CALL GP_READ (32_REG(2), "DISPLACE" 0)
1538 33 * 2 CALL GET_AREG (U(A), REPEAT_ACCUMULATOR)
1539 * 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(2) ..ONES COMPLEMENT ADDITION.
1541 * 4 UPDATE_FIXED_POINT_OVERFLOW BIT IN ASR (BIT 3)
1542 36 * 5 CALL PUT_AREG (U(A)+1, REPEAT_ACCUMULATOR)
1543 * 6 RETURN
*****

```

MSBC

AM/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 82

_LMA --LOAD NEGATIVE, FMT I, F-16

REF
PAGE

1545
1546
1547
1548

14 * 1 CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE" 0)
* 2 REPEAT_ACCUMULATOR += - REPEAT_ACCUMULATOR ..ONES COMPLEMENT.
36 * 3 CALL PUT_AREG (UCA), REPEAT_ACCUMULATOR
* 4 RETURN
*

MSWC

AM/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 03

-LM ..LOAD MAGNITUDE, FMT I, F=17

REF
PAGE

1550
1551
1552
1553
1554
1555

.....
14 * 1 CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE= 0)
* 2 IF REPEAT_ACCUMULATOR < 0, THEN
* 3 REPEAT_ACCUMULATOR := - REPEAT_ACCUMULATOR .. ONES COMPLEMENT.
* 4 ENDIF
36 * 5 CALL PUT_AREG(UCA), REPEAT_ACCUMULATOR
* 6 RETURN
.....

MSMC AM/UYK-7 (CP)
CP INSTRUCTION SET

_LB ..LOAD B, FMT I F=20

```

1557
1558
1559
1560
1561
1562
REF
PAGE
*****
* 1 .. Y -> B(U(A))(15)
* 2 CALL OP_READ(32_REG(1), "DISPLACE=0)
* 3 CALL GET_OREG(U(A), 32_REG(2))
* 4 32_REG(1)(19,3) := 32_REG(2)(19,3)
* 5 CALL PUT_OREG(U(A), 32_REG(1))
* 6 RETURN
*
*****

```

NSMC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 85

--AB --ADD B, RMT I F - 21

REF
PAGE

```

1564      * 1      .. BUI(A)(15) * Y -> BUI(A)(15)
1565      * 2      CALL CP_READ(32_REG(1), "DISPLACE=0)
1566      * 3      CALL GET_BREG(U(A), 32_REG(2))
1567      * 4      32_REG(3) := REPEAT_ACCUMULATOR(15)
1568      * 5      32_REG(3) := 32_REG(3) + 32_REG(1) --ONES COMPLEMENT ADDITION.
1569      * 6      REPEAT_ACCUMULATOR(15) := 32_REG(3)
1570      * 7      CALL PUT_BREG(U(A), REPEAT_ACCUMULATOR)
1571      * 8      BEIGH

```

MSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 86

_AND ..SUBTRACT B, FMT I, F=22

REF

PAGE

1573
1574
1575
1576
1577
1578
1579
1580

```
*****  
* 1 .. B(U(A))(15)-Y -> B(U(A))(15)  
* 2 CALL DP_READ (32_REG(1), "DISPLACE=0)  
* 3 CALL GET_BREG (U(A), REPEAT_ACCUMULATOR)  
* 4 32_REG(3) := REPEAT_ACCUMULATOR(15)  
* 5 32_REG(3) := 32_REG(3) - 32_REG(1) ..ONES COMPLEMENT SUBTRACTION.  
* 6 REPEAT_ACCUMULATOR(15) := 32_REG(3)  
* 7 CALL PUT_BREG(U(A), REPEAT_ACCUMULATOR)  
* 8 RETURN  
*****
```

MSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_SB --STORE B FMT I, F=23

```

1582
1583
1584
1585
1586
1587
REF
PAGE
*****
* 1 .. BIV(A)I(15) -> Y
* 2 CALL GET_BREG(U(A), REPEAT_ACCUMULATOR)
* 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR(15)
* 4 CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE= 0) --REPEAT_ACCUMULATOR -> MEMORY.
* 5 RETURN
*
*****

```

NSWC
AN/UYK-7 (CP)
CP INSTRUCTION SET

_SA ..STORE A FMT I, F=24

```

*****
REF *****
PAGE *****
# *****
* .. A(U(A)) -> Y
* 1 *****
* 2 CALL GET_AREGU(A), REPEAT_ACCUMULATOR)
1590 *
* 3 CALL OP_STORE(REPEAT_ACCUMULATOR, -DISPLACE= 0)
1591 *
* 4 RETURN
1592 *
*****

```

MSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_SKB ..STORE A & INDEX B FMT 1, F=25

1594	REF	*****	.. A(U(A)) -> Y, B(U(B)) := B(U(B))+1
1595	PAGE	*****	SET SPR_PRIVILEGED_INSTRUCTION_INDICATOR
1596		*****	CALL _SA
1597		*****	CALL GET_BREG(U(B), 32_REG(1))
1598		*****	32_REG(1)(15) := 32_REG(1)(15)+1
1599		*****	CALL PUT_BREG(U(B), 32_REG(1))
1600		*****	RETURN

NSWC

AN/UYN-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 90

--SMA --STORE NEGATIVE FMT I, F=26

REF
PAGE

```
1602 * 1 -- (.NOT.AUC(A)) --> Y  
1603 * 2 CALL GET_AREG(U(A)), REPEAT_ACCUMULATOR  
1604 * 3 REPEAT_ACCUMULATOR := .NOT. REPEAT_ACCUMULATOR  
1605 * 4 CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE=0")  
1606 * 5 RETURN  
*
```

MSWC

AN/UUK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 91

_SM ..STORE MAGNITUDE FMT 1, F=27

REF
PAGE

1608
1609
1610
1611
1612
1613

* 1 CALL GET_ARG(UKA),REPEAT_ACCUMULATOR)
* 2 IF REPEAT_ACCUMULATOR(31,1) = 1, THEN
* 3 REPEAT_ACCUMULATOR 1= .NOT. REPEAT_ACCUMULATOR
* 4 ENDIE
* 5 CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE=0)
* 6 RETURN

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 92

-B2 (_B5) --CLEAR & SET BIT, FMT I, F=32 & F=33

REF
PAGE

```

1615 * 1 32_REG(2) := U(X) --SAVE K FIELD & FORCE FULL WORD.
1616 * 2 U(X) := FULL WORD TYPE
1617 * 3 CALL OP_READ(REPEAT_ACCUMULATOR, "DISPLACE=0")
1618 * 4 IE U(F)_EVEN, THEN --CLEAR BIT INSTRUCTION.
1619 * 5 REPEAT_ACCUMULATOR(32_REG(2)(4),1) := 0
1620 * 6 ELSE --SET BIT INSTRUCTION.
1621 * 7 REPEAT_ACCUMULATOR(32_REG(2)(4),1) := 1
1622 * 8 ENDIE
1623 * 9 CALL_REPLACE(REPEAT_ACCUMULATOR) --UPDATE IN MEMORY.
1624 * 10 U(X) := 32_REG(2)(2)
1625 * 11 RETURN

```

MSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_RA (C_RAN) ..REPLACE ADD (SUBTRACT), FMT I, F=34 & F=36

```

REF
PAGE *****
1627 14 * 1 CALL DP_READ (REPEAT_ACCUMULATOR, "DISPLACE=0) ..FETCH OPERAND (Y) UNDER K DESIGNATOR CONTROL.
1629 33 * 2 CALL GET_ARG (U(A), 32_REG(1))
1630 * 3 IE U(F)=0'34" ..REPLACE ADD", THEN
1631 * 4 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(1) ..ONES COMPLEMENT ADDITION.
1633 * 5 ELSE ..REPLACE SUBTRACT (F=36).
1634 * 6 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR - 32_REG(1) ..ONES COMPLEMENT SUBTRACTION.
1636 * 7 ENDIE
1637 * 8 UPDATE FIXED POINT OVERFLOW (ASR(3,1))
1638 * 9 CALL UPDATE_REPLACE (REPEAT_ACCUMULATOR, U(A)+1)
1639 * 10 RETURN
*****

```

NSWC

AN/UUK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 94

_RI (C _RD) ..REPLACE INCREMENT (DECREMENT), FMT I, F=35 & F=37

REF
PAGE

```

1641 14 * 1 CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE=0")
1642 * 2 IE U(F)=0'35, "REPLACE INCREMENT", THEN
1643 * 3 REPEAT_ACCUMULATOR += REPEAT_ACCUMULATOR + 1 ..ONES COMPLEMENT.
1644 * 4 ELSE .."REPLACE DECREMENT".
1645 * 5 REPEAT_ACCUMULATOR += REPEAT_ACCUMULATOR - 1 ..ONES COMPLEMENT.
1646 * 6 ENDIE
1647 * 7 UPDATE FIXED POINT OVERFLOW (ASR(3,1))
1648 * 8 CALL UPDATE_REPLACE (REPEAT_ACCUMULATOR, U(4))
1649 * 9 RETURN

```

AN/UYK-7 (CPI)
CP INSTRUCTION SET

--MULTIPLY A, FMT I, F=4C

```

1651  * 1  A(U(A))Y -> A(U(A)+1) & A(U(A))
1652  * 2  CALL GET_AREG (U(A), 32_PEG(2)) --GET MULTIPLICAND.
1653  * 3  CALL CP_READ (32_PEG(1), "DISPLACE=0") --GET MULTIPLIER.
1654  * 4  SIGN_INDICATOR := 32_PEG(1)(31>1) .XOR. 32_PEG(2)(31>1) --REMEMBER SIGN FOR RESULT.
1655  * 5  CALL CD_MULTIPLY --DO ACTUAL MULTIPLICATION.
1656  * 6  IE SIGN_INDICATOR "NEGATIVE", THEN --NEGATE ANSWER.
1657  * 7  REPEAT_ACCUMULATOR := NOT.REPEAT_ACCUMULATOR; 32_REG(1) := NOT.32_PEG(1)
1658  * 8  ENQIE
1659  * 9  CALL PUT_AREG (U(A)+1, REPEAT_ACCUMULATOR) --STORE MOST SIGNIFICANT BITS.
1660  * 10 CALL PUT_AREG (U(A), 32_REG(1)) --STORE LEAST SIGNIFICANT BITS.
1661  * 11 RETI8B
1662  *
1663  *

```

MSMC AN/UYK-7 (CPI)
CP INSTRUCTION SET

DO_MULTIPLY

```

1665 * 1  .. MULTIPLY 32_REG(2) BY 32_REG(1) WITH RESULT IN REPEAT_ACCUMULATOR & 32_REG(1),
1666 * 2  --USING THE UYK-7 ALGORITHM (AS PER "UYK-7 LEARNER'S GUIDE (JAN 1973) PAGE 429) -
1667 * 3  FORCE 32_REG(1) POSITIVE & 32_REG(2) NEGATIVE USING ONES COMPLEMENT OPERATIONS.
1668 * 4  REPEAT_ACCUMULATOR := 0; MB_CARRY := 0 "OFF"-; COUNT := 16
1669 * 5  DO WHILE COUNT > 0
1670 * 6  DO CASE 32_REG(1)(1,2)
1671 * 7  \ON
1672 * 8  IF MB_CARRY, THEN
1673 * 9  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR-32_REG(2); MB_CARRY := 0
1674 * 10  --ONES COMPLEMENT SUBTRACT, END AROUND BORROW FORCED.
1675 * 11  ENDIE
1676 * 12  32_REG(1):=32_REG(1)-RL-2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)
1677 * 13  REPEAT_ACCUMULATES := REPEAT_ACCUMULATOR.RL-2
1678 * 14  \ON
1679 * 15  IF MB_CARRY, THEN
1680 * 16  REPEAT_ACCUMULATOR:=REPEAT_ACCUMULATOR-(32_REG(2)-LC-1); MB_CARRY:=0
1681 * 17  --ONES COMPLEMENT SUBTRACT, END AROUND BORROW FORCED.
1682 * 18  ELSE
1683 * 19  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR+(32_REG(2)) --ONES COMPLEMENT SUBTRACT.
1684 * 20  ENDIE
1685 * 21  32_REG(1):=32_REG(1)+RL-2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)
1686 * 22  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL-2
1687 * 23  \ON
1688 * 24  IF MB_CARRY, THEN
1689 * 25  ACCUMULATOR_SIGN := REPEAT_ACCUMULATOR(31,1)
1690 * 26  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR+32_REG(2) --ONES COMPLEMENT ADD.
1691 * 27  32_REG(1):=32_REG(1)+RL-2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)
1692 * 28  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL-2
1693 * 29  IF ACCUMULATOR_SIGN "NEGATIVE", THEN
1694 * 30  REPEAT_ACCUMULATOR(31,2) := 0*11' --SET TWO HIGH BITS ON.
1695 * 31  ENDIE
1696 * 32  ELSE
1697 * 33  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR-132_REG(2)-LC-1)
1698 * 34  ONES COMPLEMENT SUBTRACT, END AROUND BORROW FORCED.
1699 * 35  SUBTRACT END AROUND BORROW FORCED.
1700 * 36  32_REG(1):=32_REG(1)-RL-2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)
1701 * 37  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL-2
1702 * 38  IF -NOT-(END AROUND BORROW), THEN
1703 * 39  REPEAT_ACCUMULATOR(31,2):= 0*01' --FORCE HIGH BITS TO 0 & 1.
1704 * 40  ENDIE
1705 * 41  ENDIE
1706 * 42  \ON
1707 * 43  IF MB_CARRY, THEN
1708 * 44  32_REG(1):=32_REG(1)-RL-2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)
1709 * 45  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL-2
1710 * 46  ELSE
1711 * 47  MB_CARRY := 1 "ON"
1712 * 48  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR+32_REG(2) --TWOS COMPLEMENT ADD.
1713 * 49  32_REG(1):=32_REG(1)+RL-2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)
1714 * 50  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL-2
1715 * 51  ENDIE
1716 * 52  ENDDC --END CASE.
1717 * 53  COUNT := COUNT-1
1718 * 54

```

DO_DIVIDE (REF 64_REG, 32_REG)

REF
PAGE

```

*****
* 1  .. DIVIDE THE CONTENTS OF 64_REG BY THE CONTENTS OF 32_REG WITH THE QUOTIENT LEFT IN
* 2  ..64_REG(31,32) AND THE REMAINDER LEFT IN 64_REG(63,32). THIS USES THE UYK-7 ALGORITHM
* 3  ..AS FOUND IN THE UYK-7 LEARNER'S GUIDE (JAN 1973) PAGES 435-442.
* 4  .. (1) 64_REG - 64-BIT REFERENCE REGISTER.
* 5  .. (2) 32_REG - 32-BIT DIVISOR.
* 6  IE 32_REG = -0, THEN
* 7  32_REG = 0 ..NEGATIVE ZERO IS A SPECIAL CASE.
* 8  ENDIE
* 9  SIGN_IND := 64_REG(63,1).XOR.32_REG(1,1) ..SIGN CORRECTION FLAG.
* 10 SIGN_DIVIDEND := 64_REG(63,1) ..SAVE SIGN OF ORIGINAL DIVIDEND.
* 11 FORCE 64_REG "DIVIDEND" AND 32_REG "DIVISOR" POSITIVE (ONES COMPLEMENT)
* 12 COUNT := 32
* 13 DO WHILE COUNT > 0
* 14   64_REG := 64_REG.LL.1
* 15   CALL DIVIDE_COMPARE(64_REG,32_REG)
* 16   COUNT := COUNT-1
* 17   ENDDO ..END WHILE.
* 18 IE 64_REG(31,1) "QUOTIENT NEGATIVE", THEN ..OVERFLOW HAS OCCURRED.
* 19   ASR(3,1) "OVERFLOW" := 1
* 20 ELSE
* 21   ASR(3,1) "OVERFLOW" := 0
* 22 ENDIE
* 23 IE SIGN_IND, THEN
* 24   64_REG(31,32) := .NOT.64_REG(31,32) ..CORRECT SIGN OF QUOTIENT.
* 25 ENDIE
* 26 IE SIGN_DIVIDEND, THEN
* 27   64_REG(63,32) := .NOT.64_REG(63,32) ..MAKE REMAINDER THE SAME SIGN AS THE ORIGINAL DIVIDEND.
* 28 ENDIE
* 29 RETURN
*****

```

MSWC

AN/UUK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 98

_D ..DIVIDE A FMT I, F=41

REF
PAGE

```
1761 * 1 .. (A(U(A+1)),A(U(A))) / V -> A(U(A)) REMAINDER -> A(U(A+1))
1762 * 2 CALL OP_READ(32,REG(1),-DISPLACE=0) ..GET DIVISOR.
1763 * 3 CALL GET_AREG(U(A),64,REG(1)(31,32)) ..GET 64-BIT DIVIDEND.
1764 * 4 CALL GET_AREG(U(A+1),64,REG(1)(63,32))
1765 * 5 CALL DO_DIVIDE (64,REG(1), 32,REG(1)) ..DO THE ACTUAL DIVISION.
1766 * 6 CALL PUT_AREG(U(A+1),64,REG(1)(63,32)) ..STORE REMAINDER.
1767 * 7 CALL PUT_AREG(U(A),64,REG(1)(31,32)) .. STORE QUOTIENT.
1768 * 8 RETURN
*****
```

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 99

_BC ..COMPARE BIT TO ZERO, FMT I F=42

REF
PAGE

```

1770 * 1 J:= UCAK(4) ..IGNORE BIT 25 -> BIT 0 IS MODULO 32.
1771 * 2 UCAK := 3 ..FORCE FULL WORD TYPE OPERAND FETCH.
1772 * 3 CALL CP-READ(32_REG(1),-DISPLACEMENT=0)
1773 * 4 IF 32_REG(1)(J,1) = 0 "BIT CLEAR", THEN
1774 * 5 ASR(2,1) := 1 ..INDICATE EQUAL.
1775 * 6 ELSE ..BIT SET.
1776 * 7 ASR(2,1) := 0 ..INDICATE NOT EQUAL.
1777 * 8 ENDIE
1778 * 9 UCAK(4) := J ..RESTORE THE ORIGINAL AK FIELD.
1779 * 10 RETURN

```

- CMI .. CGMPARE INDEX INCREMENT, FMT I F=43

```

*****
REF PAGE *****
1781          34 * 1 CALL GET_BREGTU(A),32_REG(11)
1782          14 * 2 CALL CP_READ(32_REG(2),-DISPLACEMENT*O)
1783          178 * 3 IE 32_REG(11,15,16) >= 32_REG(2), THEN
1784          178 * 4 ASR(C) := 1 --INDICATE OUTSIDE LIMITS.
1785          178 * 5 32_REG(11,15,16) := 0 --CLEAR THE INDEX REGISTER.
1786          178 * 6 ELSE
1787          178 * 7 ASR(C) := 0 --INDICATE WITHIN LIMITS.
1788          178 * 8 32_REG(11,15,16) := 32_REG(11,15,16) + 1 --INCREMENT THE INDEX REGISTER.
1789          178 * 9 ENDIE
1790          37 * 10 CALL PUT_BREGTU(A),32_REG(11)
1791          179 * 11 BEIHEN
*****

```

MSDC AN/UYK-7 (CP)
CP INSTRUCTION SET

_C ..COMPARE, FMT I, F=44

```

1793  REF
1794  PAGE
1795  .. COMPARE ALU(A) : OPERAND. SET COMPARE DESIGNATORS IN ASR.
1796  C.. GET_AREG (U(A), 32_REG(1))
1797  CALL OP_READ (32_REG(2), "DISPLACE=0")
1798  CALL SET_CD1 (32_REG(1), 32_REG(2))
1799  RETURN

```

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_CL ..COMPARE LIMITS, FMT I, F=45

```
REF 1799
PAGE 1800
* 1 .. A(UCA), A(UCA)+1 ; OPERAND. SET COMPARE DESIGNATORS IN ASR.
* 2 CALL GET_AREG (UCA), 32_REG(2))
* 3 CALL GET_AREG (UCA)+1, 32_REG(3))
* 4 CALL OP_READ (32_REG(1), "DISPLACE=0)
* 5 CALL SET_CD2 (32_REG(1), 32_REG( 1), 32_REG(3))
* 6 RETURN
* ..
```

NSWC AN/UUK-7 (CP)
CP INSTRUCTION SET

_CM ..COMPARE MASKED, FMT 1, F=46

REF	PAGE	*****
1606	*	1 .. A(U(A))+1: 1 (A(U(A)).AND.OPERAND). SET COMPARE DESIGNATORS IN ASR.
1607	33 *	2 CALL GET_AREG (UCA)+1, 32_REG(1))
1608	33 *	3 CALL GET_AREG (UCA), 32_REG(2))
1609	14 *	4 CALL OP_READ (32_REG(3), "DISPLACE=0)
1610	*	5 32_REG(2) := 32_REG(2) .A. 32_REG(3)
1611	29 *	6 CALL SET_CD1 (32_REG(1), 32_REG(2))
1612	*	7 RETURN
	*	*****

NSWC

AN/UYK-7 (CPI)
CP INSTRUCTION SET

14 DEC 79 PAGE 104

CG --COMPARE GATED, FMT I, F=47

REF
PAGE

1814
1815
1816
1817
1818
1819
1820

```

* 1  .. ABS(OPERAND - A(U(A))) * A(U(A)+1). SET COMPARE DESIGNATORS IN ASR.
* 2  CALL OP_READ (32_REG(1), "DISPLACE=0")
* 3  CALL GET_AREG (U(A), 32_REG(3))
* 4  32_REG(1) := ABSOLUTE VALUE OF (32_REG(1) - 32_REG(3))
* 5  CALL GET_AREG (U(A)+1, 32_REG(2))
* 6  CALL SET_CD1 (32_REG(1), 32_REG(2))
* 7  RETURN
*

```

MSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 105

_JEP (_JOP) --JUMP EVEN (ODD) PARITY, FMT II, F=50 0 & F=50 1

```

1822      * 1 SET SPR_PRIVILEGED_INSTRUCTION_INDICATOR --INDICATE PRIVILEGED IF SPR(16,1) SET AND U(1) SET.
1824      * 2 CALL JUMP_ADDRESS (-DISPLACE-0, OPERAND_S, OPERAND_DISPLACEMENT)
1825      * 3 --DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS. OPERAND BASE
1826      * 4 --REGISTER AND OPERAND DISPLACEMENT ARE RETURNED.
1827      * 5 CALL GET_AREG (U(A), 32_REG(1))
1828      * 6 CALL GET_AREG (U(A)+1, 32_REG(2))
1829      * 7 32_REG(1) := 32_REG(1) -A. 32_REG(2)
1830      * 8 I := 0
1831      * 9 J := 0
1832      * 10 DO _WHILE I < 32
1833      * 11   J := J + 32_REG(1)(I,1)
1834      * 12   I := I + 1
1835      * 13 ENDDO
1836      * 14 J := J -X. U(F2)
1837      * 15 IF J(0) = 0, THEN --TEST FOR JOP/ODD OR JEP/EVEN
1838      * 16   CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)
1839      * 17 ENDIE
1840      * 18 RETURN

```

NSWC

ANJUYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 106

_DJZ --JUMP DOUBLE PRECISION ZERO, FMT III F-50 2

REF
PAGE

1842 * 1
1844 * 2
1845 * 3
1846 * 4
1847 * 5
1848 * 6
1849 * 7
1850 * 8
1851 * 9
*

```
*****
* 1 SET SPR_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPR(16,1) SET AND U(1) SET.
* 2 CALL JUMP_ADDRESS("DISPLACE=0, OPERAND_5, OPERAND_DISPLACEMENT)
* 3 --DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
* 4 CALL GET_AREG(U(1),64,REG(1))(31,32)
* 5 CALL GET_AREG(U(1)+1,64,REG(1))(63,32)
* 6 IF 64_REG(1) = 0, THEN --JUMP ON POSITIVE ZEPG ONLY:
* 7 CALL DO_JUMP (OPERAND_5, OPERAND_DISPLACEMENT)
* 8 ENDIE
* 9 RETURN
*****
```

NAME: 4867-7 (CP)
 OF INSTRUCTION SET

14 DEC 74 PAGE 107

--JNZ --JUMP DOUBLE PRECISION NOT ZERO, FMT III P-50 3

```

REF
PAGE .....
1053 * 1 SET SPB_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPB(16,1) SET AND U(1) SET.
1055 * 2 CALL JUMP_ADDRESS("DISPLACE=0, OPERAND_S", OPERAND_DISPLACEMENT)
1056 * 3 --DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
1057 * 4 CALL GET_AREG(U(4),64,REG(1)(31,32))
1058 * 5 CALL GET_AREG(U(4)+1,64,REG(1)(63,32))
1059 * 6 IF 64_REG(1) <> 0, THEN --TEST POSITIVE ZERO ONLY
1060 * 7 CALL DC_JMP (OPERAND_S, OPERAND_DISPLACEMENT)
1061 * 8 ENDIE
1062 * 9 BEIUBB
.....

```

_F51C_JP, _JN, _J7, _JN2, FMT III F-51 O YHRU 51 3

```

1864 * 1 SET SPB_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPB(16,1) SET AND UC1) SET.
1865 * 2 CALL JUMP_ADDRESS--DISPLACE*0, OPERAND_S, OPERAND_DISPLACEMENT)
1866 * 3 --DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
1867 * 4 CALL GET_AREGU(A),32(REG1))
1868 * 5 JUMP := FALSE
1869 * 6 DO _CASE UC(3) OF
1870 * 7   \0\ IF 32(REG1)(31,1) = 0 "POSITIVE", THEN JUMP := TRUE ENDOF
1871 * 8   \1\ IF 32(REG1)(31,1) = 1 "NEGATIVE", THEN JUMP := TRUE ENDOF
1872 * 9   \2\ IF 32(REG1) = 0 "POSITIVE ZERO", THEN JUMP := TRUE ENDOF
1873 * 10  \3\ IF 32(REG1) <> 0 "NOT POSITIVE ZERO", THEN JUMP := TRUE ENDOF
1874 * 11 ENDDO
1875 * 12 IF JUMP := TRUE
1876 * 13   CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)
1877 * 14 ENDAE
1878 * 15 RETURN
1879 *

```

MSHC AN/UYK-7 (CP)
CP INSTRUCTION SET

_LBJ ..LOAD B AND JUMP, FMT III, F=52 C

1881	REF	*****	SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.
1883	PAGE	*****	
1884		1	IF U(A) = 0, THEN
1885		2	RETURN ..NO OPERATION.
1886		3	ELSE
1887		4	32_REG(1)(19,3) := P(5) ..P REGISTER BASE REGISTER DESIGNATOR.
1888		5	32_REG(1)(15,16) := P(6) ..P REGISTER DISPLACEMENT (ALREADY INCREMENTED).
1889		6	CALL PUT_REG(U(A),32_REG(1))
1890	37	7	CALL JUMP_ADDRESS(DISPLACE0, OPERAND_S, OPERAND_DISPLACEMENT)
1891	13	8	CALL JUMP_ADDRESS(DISPLACE0, OPERAND_S, OPERAND_DISPLACEMENT)
1892		9	..OO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
1893		10	CALL OO_JUMP (COPERAND_S, OPERAND_DISPLACEMENT)
1894	23	11	ENDIE
		12	RETURN

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 110

_JBMZ --INDEX JUMP B, FMT III, F=52 I

REF

PAGE

```

1896 * 1
1898 * 2 SET SPR_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.
1899 * 3 CALL GET_BREG(U(A),32(REG(I)))
1900 * 4 IE 32(REG(I))(15,16) <> 0, THEN
1902 * 5 32_REG(I)(15,16) := 32_REG(I)(15,16) - 1 --DECREMENT THE INDEX REGISTER.
1903 * 6 CALL PUT_BREG(U(A),32(REG(I)))
1904 * 7 CALL JUMP_ADDRESS(=DISPLACE=0, OPERAND_5, OPERAND_DISPLACEMENT)
1905 * 8 --DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
1906 * 9 CALL DO_JUMP (OPERAND_5, OPERAND_DISPLACEMENT)
1907 * 10 ENDIE
      RETURN

```

MSBC AN/UYK-7 (CP)
CP INSTRUCTION SET

_JS --JUMP SY & B, FMT III F=52 Z

1909	REF	*****	
1910	PAGE	*****	
1911		*****	
1912		*****	
1913		*****	
1914		*****	
1915		*****	
1916		*****	

34	*	1	CALL GET_BREG(UB), 32_REG(1)
35	*	2	P(S) := 32_REG(1)(19,3) --P(S) GETS NEW BASE REGISTER DESIGNATOR.
36	*	3	P(D) := 32_REG(1)(19) + U(SV) --P(D) GETS NEW DISPLACEMENT.
37	*	4	32_REG(1) := P(D)
38	*	5	CALL ADD_SIP(S), 32_REG(1) --JUMPED TO ADDRESS GOES TO 32_REG(1).
39	*	6	CALL OPP_CHECK(32_REG(1), OPERAND) --DO OPERAND CHECK ON JUMPED TO ADDRESS.
40	*	7	RETURN
41	*		*****

_JL --JUMP LOWER, FMT III, F-92 3

REF
PAGE

```
1918 * 1 SET SPR_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPR(10,1) SET AND U(1) SET.
1920 * 2 -- NOTE: BPR CHECKS & ILLEGAL INSTRUCTION CHECKS ARE DONE IN THIS ROUTINE TO
1922 * 3 --SUPPLEMENT NORMAL I SEQUENCE HANDLING OF THE LOWER HALF WORD INSTRUCTION.
1924 * 4 CALL JUMP_ADDRESS("DISPLACE", OPERAND_0, OPERAND_5, OPERAND_DISPLACEMENT)
1925 * 5 --DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
1926 * 6 P(5) := OPERAND_5 --SET UP BASE (5) DESIGNATOR FOR JUMPED TO INSTRUCTION.
1928 * 7 P(0) := OPERAND_DISPLACEMENT --SET UP DISPLACEMENT FOR JUMPED TO INSTRUCTION.
1930 * 8 32_REG(1) := P(0) --COPY THE DISPLACEMENT FOR CHECKS.
1931 * 9 ASR(15,1) := 1 --SET LOWER HALF WORD INDICATOR.
1932 * 10 CALL ADD_S(UCS), 32_REG(1)) --COMPUTE ABSOLUTE ADDRESS OF JUMPED TO INSTRUCTION.
1934 * 11 CALL BPR_CHECK(32_REG(1), INSTRUCTION)
1935 * 12 --DO BPR CHECKS, PECULIAR TO LOWER HALF WORD INSTRUCTIONS ARRIVED AT BY JL INSTRUCTION.
1937 * 13 CALL MEMORY_READ(32_REG(1), 32_REG(2), UCS), INSTRUCTION) --FETCH INSTRUCTION.
1939 * 14 -- NOTE: THE INITIAL CALL TO OP_READ INSURES THE INSTRUCTION IS AT A
1940 * 15 --VALID MEMORY LOCATION, THUS NO INTERRUPT SHOULD OCCUR.
1941 * 16 IE 32_REG(2)(15,2) <> B'11', THEN --CHECK FOR VALID HALF-WORD INSTRUCTION CANDIDATE.
1943 * 17 CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-1", OP_ILLEGAL_INSTRUCTION)
1944 * 18 END
1945 * 19 RETURN
*
```

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 113

_J530 --JUMP ON ASR OVERFLOW DESIGNATOR (JMF, JOF), FMT III F-93 0

REF
PAGE

1947
1949
1950
1951
1952
1953
1954
1955
1956

```
* 1 SET SPR_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPR(16,1) SET AND U(1) SET.
* 2 CALL JUMP_ADDRESS("DISPLACE",G, OPERAND_S, GPERAND_DISPLACEMENT)
* 3 --DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS..
* 4 J := ASR(3,1) "OUTSIDE/WITHIN" X. U(A)
* 5 ASR(3,1) := 0 --CLEAR ASR OVERFLOW INDICATOR.
* 6 IF J(O)=0, THEN
* 7     CALL DO_JUMP(OPERAND_S, OPERAND_DISPLACEMENT)
* 8 ENDIE
* 9 RETURN
*
```

_J531 ..JUMP ON ASR COMPARE DESIGNATOR, FMT 1:: F-53 1

REF
PAGE

```

1958 * 1 .. THIS ROUTINE EXECUTES THE FOLLOWING INSTRUCTIONS : JNE, JE, JG, JGE, JLT, JLE, JNW, JW.
1960 * 2 SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(1) SET.
1962 * 3 CALL JUMP_ADDRESS=DISPLACE=0, OPERAND_S, OPERAND_DISPLACEMENT)
1963 * 4 ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS
1964 * 5 _JUMP := FALSE ..INITIALIZE THE JUMP FLAG.
1965 * 6 DO CASE U(A) OF
1966 * 7 .. V(A) V(A) ..JNE, JE.
1967 * 8 J := U(A) ..X. ASR(2,1) "EQUAL/UNEQUAL"
1968 * 9 IE J(0,1)=0, THEN
1969 * 10 _JUMP := TRUE
1970 * 11 ENDIE
1971 * 12 .. V(A) ..JG.
1972 * 13 IE ASR(1,1)=1 "GREATER THAN OR EQUAL" ..A. ASR(2,1)=0 "UNEQUAL", THEN
1973 * 14 _JUMP := TRUE
1974 * 15 ENDIE
1975 * 16 .. V(A) V(A) ..JGE, JLT.
1976 * 17 J := U(A) ..X. ASR(1,1) "GREATER THAN OR EQUAL/LESS THAN"
1977 * 18 IE J(0,1)=0, THEN
1978 * 19 _JUMP := TRUE
1979 * 20 ENDIE
1980 * 21 .. V(A) ..JLE.
1981 * 22 IE ASR(1,1)=0 "LESS THAN" ..V. ASR(2,1)=1 "EQUAL", THEN
1982 * 23 _JUMP := TRUE
1983 * 24 ENDIE
1984 * 25 .. V(A) V(A) ..JNW, JW.
1985 * 26 J := U(A) ..X. ASR(0,1) "OUTSIDE/WITHIN"
1986 * 27 IE J(0,1)=1, THEN
1987 * 28 _JUMP := TRUE
1988 * 29 ENDIE
1989 * 30 ENDDO
1990 * 31 IE _JUMP = TRUE, THEN
1991 * 32 CALL DO_JUMP(OPERAND_S, OPERAND_DISPLACEMENT)
1992 * 33 ENDIE
1993 * 34 RETURN

```

MSWC AN/UYK-7 (ICP)
CP INSTRUCTION SET

_J532 ..RETURN JUMPS, FMT III F=53 2

```

1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028

REF
PAGE
*****
* 1 SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPP(16,1) SET AND U(1) SET.
* 2 _MATCH := FALSE ..INITIALIZE THE MATCH FLAG.
* 3 JUMP_STOP := U(A)(2,1) ..HIGH ORDER BIT OF "A" FIELD.
* 4 IE JUMP_STOP .AND. ASR(19,4)=0 "TASK MODE", THEN
* 5 GENERATE_SYNCHRONOUS_INTERRUPT("P=1", PRIVILEGED_INSTRUCTION_VIOLATION)
* 6 .. ABORTS THE INSTRUCTION.
* 7 ENDIE
* 8 IE U(A)(1,2), THEN ..CHECK APPROPRIATE JUMP SWITCH.
* 9 IE JUMP_STOP, THEN
* 10 IE STOP_SWITCH(U(A)) "-1", THEN
* 11 _MATCH := TRUE
* 12 ENDIE
* 13 ELSE
* 14 IE JUMP_SWITCH(U(A)) "-1", THEN
* 15 _MATCH := TRUE
* 16 ENDIE
* 17 ELSE
* 18 ..0 -> ALWAYS JUMP
* 19 _MATCH := TRUE
* 20 ENDIE
* 21 32_REG(1)(31,15) := P(15)
* 22 32_REG(1)(15) := P(0)
* 23 CALL OP_STORE(32_REG(1), "DISPLACE=0", "STORE P.
* 24 CALL JUMP_ADDRESS("DISPLACE=1, OPERAND_S, OPERAND_DISPLACEMENT)
* 25 ..OO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
* 26 IE JUMP_STOP .OR. (_MATCH = TRUE), THEN
* 27 CALL DO_JUMP(OPERAND_S, OPERAND_DISPLACEMENT)
* 28 IE JUMP_STOP .AND. (_MATCH = TRUE), THEN
* 29 SUSPEND_PROCESSING (INDICATING STOP ACTIVE) UNTIL RESTARTED
* 30 ENDIE
* 31 ENDIE
* 32 RETURN
*****

```

NSWC

AN/DYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 116

J533 --MANUAL JUMPS, FMT III, F-93 3

REF
PAGE

```
*****
* 1 SET SPR_PRIVILEGED_INSTRUCTION_INDICATOR
* 2 CALL JUMP_ADDRESS ("DISPLACE", OPERAND_S, OPERAND_DISPLACEMENT)
* 3 _MATCH := FALSE --ASSUME NO JUMP.
* 4 JUMP_STOP := UCA(2,1) --I.E. HIGH BIT OF A FIELD.
* 5 IF JUMP_STOP .AND. ASR(19,4) = 0 "TASK MODE", THEN
* 6 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-1", PRIVILEGED_INSTRUCTION_VIOLATION);
* 7 --ABORT THIS INSTRUCTION.
* 8 ENDIE
* 9 IF UCA(1,2), THEN --CHECK APPROPRIATE JUMP SWITCH.
* 10 IF JUMP_STOP, THEN
* 11 IF STOP_SWITCH(UCA) = "1", THEN
* 12 _MATCH := TRUE
* 13 ENDIE
* 14 ELSE
* 15 IF JUMP_SWITCH(UCA) = "1", THEN
* 16 _MATCH := TRUE
* 17 ENDIE
* 18 ELSE --0 -- ALWAYS JUMP.
* 19 _MATCH := TRUE
* 20 ENDIE
* 21 IF (JUMP_STOP .OR. (_MATCH=TRUE)), THEN
* 22 CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)
* 23 IF (JUMP_STOP .AND. (_MATCH=TRUE)), THEN
* 24 SUSPEND_PROCESSING (INDICATING_STOP_ACTIVE) UNTIL_RESTARTED
* 25 ENDIE
* 26 RETURN
* 27
* 28
*****
```

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

LCT (6 LCT) --LOAD CMR, FMT I, F-54 & F-55

2060	REF	*****	
2061	PAGE	*****	
2062		1	--TREATED AS A FORMAT III INSTRUCTION IN THAT UCK) DOESN'T AFFECT OPERAND.
2063		2	32_REG(2) := UCK) --ISOLATE AND SAVE "AK" FIELD.
2064		3	IF UCK) GOD, THEN --INTERRUPT CONTROL MEMORY REFERENCED.
2065		4	32_REG(2) := 32_REG(2)*0*100* --DISPLACE TO INTERRUPT ADDRESSES.
2066		5	ENDIE
2067		6	IF (32_REG(2)>0*17) .OR. REPEAT_IN_PROGRESS) .AND. ASR(19,4)=0 "TASK MODE", THEN
2068		7	CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P=-1, PRIVILEGED INSTRUCTION VIOLATION")
2069		8	--ABORT THIS INSTRUCTION.
2070		9	ENDIE
2071		10	UCK) := 3 "FULL WORD"
2072		11	CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE=0)
2073		12	CMR(32_REG(2)) := REPEAT_ACCUMULATOR --UPDATE CONTROL MEMORY CONTENTS.
2074		13	IF REPEAT_IN_PROGRESS, THEN
2075		14	32_REG(2) := 32_REG(2)+1 --INCREMENT AK FIELD.
2076		15	ENDIE
2077		16	UCK) := 32_REG(2)(5)
2078		17	RETURN
2079			*****

MSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 110

SCT (E SCI) --STORE CMR, FMT I, F=56 & F=57

REF
PAGE

```
2081 * 1 ..TREATED AS A FORMAT III INSTRUCTION IN THAT UICK DOESN'T AFFECT OPERAND.
2082 * 2 32_REG(2) := U(CK) --ISOLATE AND SAVE "AK" FIELD.
2083 * 3 IE U(F) ODD, THEN --INTERRUPT CONTROL MEMORY REFERENCED.
2084 * 4 32_REG(2) := 32_REG(2)+0'100' --DISPLACE TO INTERRUPT ADDRESSES.
2085 * 5 ENDIE
2086 * 6 IE (32_REG(2)>0'17' .OR. REPEAT_IN_PROGRESS) .AND. ASR(19,4)=0 "TASK MODE", THEN
2087 * 7 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P=-1, PRIVILEGED INSTRUCTION VIOLATION")
2088 * 8 --AEOR? THIS INSTRUCTION.
2089 * 9 ENDIE
2090 * 10 REPEAT_ACCUMULATOR := CHR(32_REG(2)) --CONTROL MEMORY CONTENTS GOES TO REPEAT_ACCUMULATOR.
2091 * 11 U(CK) := 3 "FULL WORD"
2092 * 12 CALL OP_STORE (REPEAT_ACCUMULATOR, "DISPLACE=0")
2093 * 13 IE REPEAT_IN_PROGRESS, THEN
2094 * 14 32_REG(2) := 32_REG(2)+1 --INCREMENT AK FIELD.
2095 * 15 ENDIE
2096 * 16 U(CK) := 32_REG(2)(5) --RESTORE AK FIELD.
2097 * 17 RETURN
2098 *
2099 *
2100 *
2101 *
```

MSC_60 ..STORE CMR IN A, FRI IV A, F=60

REF
PAGE

```

*****
* 1  ADR := U{AF4} ..EXTRACT CONTROL MEMORY ADDRESS (I.E. 6-BIT "A" & "F4" FIELDS).
* 2  IE U{1} "MSCI", THEN
* 3  ADR := ADR + 0'100' ..DISPLACE TO INTERRUPT ADDRESSES.
* 4  ENDIE
* 5  IE ADR>=0'20' .AND. ASRC(19,4)=0 "TASK MODE", THEN
* 6  CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-1", PRIVILEGED INSTRUCTION VIOLATION)
* 7  ..ABORT THIS INSTRUCTION.
* 8  ENDIE
* 9  CALL HALFWOED_TOGGLE ..CALL AFTER INTERRUPT POSSIBILITY FOR ECP 97A.
* 10 IE (ADR<=0'137' .AND. ADR>=0'130') .OR. (ADR<=0'57' .AND. ADR>=0'30'), THEN
* 11 32_REG(1) := 0 ..UNASSIGNED CONTROL MEMORY IS A SOURCE OF ZEROS.
* 12 ELSE
* 13 IE U{AF4}<=0'17' .AND. U{AF4}>=0'11' "INDEX REGISTER REFERENCE", THEN
* 14 32_REG(1) := CMR{ADR}{15} ..EXTRACT ONLY 16-BITS.
* 15 ELSE
* 16 IE ADR<=0'77' .AND. ADR>=0'70' "ACTIVE STATUS REGISTER", THEN
* 17 32_REG(1) := CMR{70} ..ALL 7X REFER TO ASR.
* 18 ELSE
* 19 IE ADR>=60 .AND. ADR<=67 "BREAKPOINT REGISTER", THEN
* 20 32_REG(1) := CMR{60} ..ALL 6X REFER TO BREAKPOINT REGISTER.
* 21 ELSE ..ALL OTHER REFERENCES.
* 22 32_REG(1) := CMR{ADR}
* 23 ENDIE
* 24 ENDIE
* 25 ENDIE
* 26 ENDIE
* 27 CALL PUT_AREG (U{8}, 32_REG(1))
* 28 RETURN
*****

```

HLC_61 ..LOAD CMR FROM A, FMT IV A, F-61

```
*****
REF PAGE
33
2135 * 1 ADR := UCAF4} ..EXTRACT CONTROL MEMORY ADDRESS (I.E. 6-BIT "A" & "F4" FIELDS).
2137 * 2 IE UC1} "HLCI", THEN
2138 * 3 ADR := ADR + 0'100' ..DISPLACE TO INTERRUPT ADDRESSES.
2139 * 4 ENDIE
2140 * 5 IE ADR>=0'20' .AND. ASR(19,4)=0 "TASK MODE", THEN
2141 * 6 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P"-1, PRIVILEGED INSTRUCTION VIOLATION)
2142 * 7 ..ABORT THIS INSTRUCTION.
2143 * 8 ENDIE
2144 * 9 CALL HALFWORD_TOGGLE ..CALL AFTER INTERRUPT POSSIBILITY FOR ECP 97A.
2145 * 10 IE (ADR<=0'137' .AND. ADR>=0'130') .OR. (ADR<=0'57' .AND. ADR>=0'30'), THEN
2146 * 11 RETURN ..NOT ACCESSIBLE, THUS NOOP.
2147 * 12 ELSE
2148 * 13 CALL GET_AREG (UC0}, 32_REG(1))
2149 * 14 IE UCAF4}<=0'17' .AND. UCAF4}>=0'11' "INDEX REGISTER REFERENCE", THEN
2150 * 15 CHR(ADR)(15) := 32_REG(1)(15) ..UPDATE ONLY LOW 16 BITS.
2151 * 16 ELSE
2152 * 17 IE ADR<=0'77' .AND. ADR>=0'70' "ACTIVE STATUS REGISTER", THEN
2153 * 18 CMR(70)(15) := 32_REG(1)(15) ..CAN ONLY CHANGE LOW 16 BITS.
2154 * 19 ELSE
2155 * 20 IE ADR>=60 .AND. ADR<=1 "BREAKPOINT REGISTER", THEN
2156 * 21 CMR(60) := 32_REG(1) ..ALL 6X REFER TO THE BREAKPOINT REGISTER.
2157 * 22 ELSE ..ALL OTHER REFERENCES.
2158 * 23 CMR(ADR) := 32_REG(1)
2159 * 24 ENDIE
2160 * 25 ENDIE
2161 * 26 ENDIE
2162 * 27 ENDIE
2163 * 28 RETURN
2164
*****
```

MSHC AN/UYK-7 (CP)
CP INSTRUCTION SET

_HLC (S _MRZ S _MRS) SHIFT LEFT CIRCULARLY (RIGHT LOGICAL S

```

2167 REF
2168 PAGE *****
2170 * 1 --RIGHT ARITHMETIC); FMT IV B, F=62, F=64, F=66
2171 * 2 CALL GET_SHIFT_AMOUNT (UCH), SHIFT_COUNT) --ACQUIRE 6-BIT SHIFT COUNT.
2172 * 3 CALL GET_AREG (UCA), 32_REG(1))
2173 * 4 DO CASE UCF) --SHIFT ACCORDING TO OPERATION CODE.
2174 * 5 162\ SHIFT 32_REG(1) LEFT CIRCULAR BY SHIFT_COUNT
2175 * 6 164\ SHIFT 32_REG(1) RIGHT LOGICAL BY SHIFT_COUNT
2176 * 7 166\ SHIFT 32_REG(1) RIGHT ARITHMETIC BY SHIFT_COUNT.
2177 * 8 ENDDO --END CASE.
2178 * 9 CALL PUT_AREG (UCA), 32_REG(1)) --RESTORE SHIFTED VALUE.
2179 * 10 CALL HALFWORD_TOGGLE
2180 * 11 RETURN
2181 *
*****

```

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 122

-MOLC (E _HDRZ E _MCRS) --SHIFT LEFT DOUBLE CIRCULAR E

REF
PAGE

2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203

```

* 1  ..PIGMENT LOGICAL E RIGHT ARITHMETIC, FRT IV 8, F=63, F=65, F=67
* 2  CALL GET_SHIFT_AMOUNT (U(A), SHIFT_COUNT) --ACQUIRE 0-BIT SHIFT COUNT.
* 3  CALL GET_AREG (U(A)+1, 64, REG(1)(63,32)) --MOST SIGNIFICANT BITS.
* 4  CALL GET_AREG (U(A), 64, REG(1)(31)) --LEAST SIGNIFICANT BITS.
* 5  DE CASE U(F) --SHIFT ACCORDING TO OPERATION CODE.
* 6      163N SHIFT 64, REG(1) LEFT CIRCULAR BY SHIFT_COUNT
* 7      165N SHIFT 64, REG(1) RIGHT LOGICAL BY SHIFT_COUNT
* 8      167N SHIFT 64, REG(1) RIGHT ARITHMETIC BY SHIFT_COUNT
* 9      ENDD --END CASE_
* 10 CALL PUT_AREG (U(A)+1, 64, REG(1)(63,32)) --MOST SIGNIFICANT BITS.
* 11 CALL PUT_AREG (U(A), 64, REG(1)(31)) --LEAST SIGNIFICANT BITS.
* 12 CALL HALFWORD_TOGGLE
* 13 RETURN
*

```

NSWC AN/UYK-7 (CPI)
CP INSTRUCTION SET

_MSF --SCALE FACTOR, FMT IN A, F=70 0

```

2194 33 * 1 CALL GET_AREG (U(A), 32_REG(1))
2195 * 2 IF 32_REG(1) < 1, THEN
2196 * 3   SHIFT_COUNT := (NUMBER OF LEFT JUSTIFIED ONE BITS IN 32_REG(1)) - 1
2197 * 4 ELSE
2198 * 5   SHIFT_COUNT := (NUMBER OF LEFT JUSTIFIED ZERO BITS IN 32_REG(1)) - 1
2199 * 6 ENDIF
2200 * 7 ..NOTE THAT ALL ONES OR ZEROS MEANS THAT SHIFT_COUNT = 0/37.
2201 * 8 IF SHIFT_COUNT < 0/37, THEN
2202 * 9   SHIFT LEFT CIRCULARLY 32_REG(1) BY SHIFT_COUNT
2203 * 10 CALL PUT_AREG (U(A), 32_REG(1)) ..RESTORE SHIFTED VALUE.
2204 * 11 ENDIF
2205 * 12 IF U(A) <> U(B), THEN
2206 * 13   CALL PUT_AREG (U(B), SHIFT_COUNT)
2207 * 14 ENDIF
2208 * 15 CALL HALFWORD_TOGGLE
2209 * 16 RETURN

```

NSWC

AN/UYN-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 124

_HDSF ..DOUBLE SCALE FACTOR, FMT 14 A, F=70 1

REF
PAGE

```
2211 33 * 1 CALL GET_AREG(U(A)+1, 64_REG(1)(63,32))
2212 33 * 2 CALL GET_AREG(U(A), 64_REG(1)(63))
2213 * 3 IF 64_REG(1)(63,1), THEN
2214 * 4 SHIFT_COUNT := (# OF LEFT JUSTIFIED ONES) - 1
2215 * 5 ELSE
2216 * 6 SHIFT_COUNT := (# OF LEFT JUSTIFIED ZEROS) - 1
2217 * 7 ENDIF
2218 * 8 IF SHIFT_COUNT < 0.77, THEN
2219 * 9 SHIFT LEFT CIRCULARLY (64_REG(1), SHIFT_COUNT)
2220 36 * 10 CALL PUT_AREG (U(A)+1, 64_REG(1)(63,32))
2221 36 * 11 CALL PUT_AREG (U(A), 64_REG(1)(63))
2222 * 12 ENDF
2223 * 13 IF U(A)<>U(B) .AND. (U(A)+1)<>U(B), THEN
2224 36 * 14 CALL PUT_AREG(U(B),SHIFT_COUNT)
2225 * 15 ENDF
2226 * 16 CALL HALFWORD_TOGGLE
2227 * 17 RETURN
*****
```

MSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_HCP ..COMPLEMENT A, FMT IV A, F-70 2

REF	PAGE	
2229	33	* 1 CALL GET_AREG(U{A}, 32_REG(1))
2230	33	* 2 32_REG(1) := .NOT. 32_REG(1)
2231	36	* 3 CALL PUT_AREG(U{A}, 32_REG(1))
2232		* 4 CALL HALFWORD_TOGGLE
2233		* 5 RETURN

NSWC ANALYZE-7 (CP)
CP INSTRUCTION SET

_MDCP .. DOUBLE COMPLEMENT A, FMT IV A, F-70 3

```

2235 33 * 1 CALL GET_AREG(U(A)+1, 32_REG(2))
2236 36 * 2 32_REG(2) := -NOT. 32_REG(2)
2237 36 * 3 CALL PUT_AREG(U(A)+1, 32_REG(2))
2238 125 * 4 CALL _MCP
2239 2239 * 5 RETURN

```

NSWC AN/UYM-7 (CPI)
CP INSTRUCTION SET

_MORE(_HA_ _HAN_ _HICP_ _HAND), FMT IV A, F-71 0 THPU 71 3 6 F-71 5

```

REF
PAGE *****
33 * 1 CALL GET_AREG(U(A), 32_REG(1))
33 * 2 CALL GET_AREG(U(S), 32_REG(2))
2241 * 3 DC _CASE U(4)
2242 * 4 10\ LOGICAL SUM
2243 * 5 32_REG(1) := 32_REG(1) .V. 32_REG(2)
2244 * 6 11\ SUM
2245 * 7 32_REG(1) := 32_REG(1) + 32_REG(2) ..ONES COMPLEMENT ADDITION.
2246 * 8 UPDATE FIXED POINT OVERFLOW INDICATOR(ASR(3,1))
2247 * 9 12\ DIFFERENCE
2248 * 10 32_REG(1) := 32_REG(1) - 32_REG(2) ..ONES COMPLEMENT SUBTRACTION.
2249 * 11 UPDATE FIXED POINT OVERFLOW INDICATOR(ASR(3,1))
2250 * 12 13\ LOGICAL DIFFERENCE
2251 * 13 32_REG(1) := 32_REG(1) .X. 32_REG(2)
2252 * 14 15\ AND
2253 * 15 32_REG(1) := 32_REG(1) .A. 32_REG(2)
2254 * 16 ENDDC .. _ENDCASE
2255 * 17 CALL PUT_AREG(U(A), 32_REG(1))
2256 * 18 CALL HALFWORD_TOGGLE
2257 * 19 RETURN
2258
*****

```

NSWC

AN/U/K-7 (CPI
CP INSTRUCTION SET

14 DEC 79 PAGE 128

_HM --MULTIPLY REGISTER, FMT IV A, F=74 0

REF
PAGE

```

2261 * 1  .. A(U(A)) X A(U(B)) -> A(U(A)+1); A(U(A))
2262 * 2  CALL GET_AREG (U(A), 32_REG(2)) --GET MULTIPLICAND.
2263 * 3  CALL GET_AREG (U(B), 32_REG(1)) --GET MULTIPLIER.
2264 * 4  SIGN_INDICATOR := 32_REG(31); X:= 32_REG(2)(31,1) --REMEMBER SIGN FOR RESULT.
2265 * 5  CALL DO_MULTIPLY --DO ACTUAL MULTIPLICATION
2266 * 6  IF SIGN_INDICATOR =NEGATIVE-, THEN --NEGATE ANSWER.
2267 * 7  REPEAT_ACCUMULATOR := .NOT. REPEAT_ACCUMULATOR
2268 * 8  32_REG(1) := .NOT. 32_REG(1)
2269 * 9  ENQIE
2270 * 10 CALL PUT_AREG (U(A)+1, REPEAT_ACCUMULATOR) --STORE MOST SIGNIFICANT BITS
2271 * 11 CALL PUT_AREG (U(A), 32_REG(1)) --STORE LEAST SIGNIFICANT BITS
2272 * 12 CALL HALFWORD_TOGGLE
2273 * 13 RETURN
2274 *

```

MSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 129

_MD ..DIVIDE REGISTER FMT IV A, F-74 1

REF
PAGE

```

2275 * 1 ..(A(U(A+1)), A(U(A))) / A(U(B)) -> A(U(A)); REMAINDER -> A(U(A+1))
2277 * 2 CALL GET_AREG(U(B), 32_REG(2)) ..GET DIVISOR.
2278 * 3 CALL GET_AREG(U(A), 64_REG(1)) (31,32) ..GET DIVIDEND.
2279 * 4 CALL GET_AREG(U(A)+1, 64_REG(1)) (63,32)
2280 * 5 CALL DO_DIVIDE (64_REG(1), 32_REG(2)) ..DO ACTUAL DIVISION.
2281 * 6 CALL PUT_AREG(U(A)+1, 64_REG(1)) (63,32) ..STORE REMAINDER.
2282 * 7 CALL PUT_AREG(U(A), 64_REG(1)) (31,32) ..STORE QUOTIENT.
2283 * 8 CALL HALFWORD_T066LE
2284 * 9 RETURN

```

MSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 130

_MRT --SQUARE ROOT FMT IV A F-74 2

```

REF
PAGE
2286 * 1 .. THIS INSTRUCTION USES THE ALGORITHMS ON PAGE 445-452 OF THE UYK-7 LEARNER'S GUIDE
2288 * 2 .. (JANUARY 1973). UPON COMPLETION 32_REG(1) EQUALS THE SQUARE ROOT AND 64_REG(1)(63:32)
2290 33 * 3 CALL GET_AREG(U(A)+1, 64_REG(1)(31:32)) --GET HIGH ORDER BITS OF RADICAND.
2292 * 4 IF 64_REG(1)(31:32) "UPPER 2 BITS OF RADICAND", THEN
2293 * 5 ASR(3,1) := 1 --INDICATE OVERFLOW.
2294 * 6 ELSE
2295 * 7 ASR(3,1) := 0 --CLEAR OVERFLOW.
2296 * 8 ENDIF
2297 * 9 64_REG(1)(63:32) := 0 --INITIALIZE THE CURRENT RESIDUE.
2298 * 10 32_REG(1) := 0 --INITIALIZE THE ROOT.
2299 * 11 CALL DO_SORT(64_REG(1), 32_REG(1)) --WORK ON HIGH ORDER BITS OF THE RADICAND.
2301 * 12 --NOTE: DO_SORT USES 32_REG(2) FOR SCRATCH.
2302 * 13 CALL GET_AREG(U(A), 64_REG(1)(31:32)) --GET LEAST SIGNIFICANT BITS RADICAND.
2304 * 14 CALL DO_SORT(64_REG(1), 32_REG(1)) --WORK ON LOW BITS OF RADICAND.
2305 * 15 32_REG(1) := 32_REG(1) .RL.1 --COMPENSATE FOR EXTRA SHIFT IN DO_SORT.
2306 * 16 CALL PUT_AREG(U(B), 32_REG(1)) --STORE ROOT.
2307 * 17 CALL PUT_AREG(U(C)+1, 64_REG(1)(63:32)) --STORE RESIDUE.
2308 * 18 CALL HALFWORD_TOGGLE
2309 * 19 RETURN
*

```

DQ_SORT (REF 64_REG, REF 32_REG)

REF
PAGE

2311
2312
2313
2315
2317
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328

```

*****
* 1 COUNT := 16
* 2 DO WHILE COUNT > 0
* 3   64_REG := 64_REG .LL. 2 --THE PARTIAL RADICAND (64_REG(63,32)) IS FORMED BY SHIFTING THE
* 4   --CURRENT RESIDUE (64_REG(63,32)) LEFT TWO BITS AND SUBSTITUTING TWO BITS FROM THE ORIGINAL
* 5   --RADICAND (64_REG(31,32)). THE PARTIAL RADICAND (64_REG(63,32)) MAY REQUIRE MORE THAN
* 6   --32 BITS TO EXPRESS.
* 7   32_REG := 32_REG .LL. 1 --SHIFT PARTIAL ROOT LEFT 1.
* 8   32_REG(2) := 32_REG + 1 --32_REG(2) IS THE 32 BIT TRIAL ROOT EXTRACTOR.
* 9   IF 32_REG >= 32_REG(2), THEN --UNSIGN 34 BIT COMPAE (TO BE SAFE).
* 10  32_REG := 32_REG - 32_REG(2) --32_REG NOW HAS THE NEW CURRENT RESIDUE.
* 11  32_REG(1,1) := 1 --SET BIT IN ROOT.
* 12  ENDIE
* 13  COUNT := COUNT-1
* 14  ENDDO
* 15  RETURN
*****

```

NSRC

AN/UYK-7 (CPI)
CP INSTRUCTION SET

14 DEC 79 PAGE 132

_HLB ..LOAD B(A) WITH B(B) FMT IV A, F-74 3

REF
PAGE

2330
2331
2332
2333
2334
2335
2336

* 1 ..B(B) -> B(U(A))
* 2 CALL GET_BREG(U(B), 32_REG(1))
* 3 CALL GET_BREG(U(A), 32_REG(2))
* 4 32_REG(2)(15) := 32_REG(1)(15)
* 5 CALL PUT_BREG(U(A), 32_REG(2))
* 6 CALL HALFWORD_TOGGLE
* 7 RETURN
*

MSWC

AN/UUK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 133

_HC ..COMPARE REGISTER, FMT IV A, F-74 4

```
REF
PAGE *****
* 1 .. A(U(A)) IS COMPARED TO A(U(B)) ; SET ASR(2,2) "CD" ACCORDINGLY.
* 2 CALL GET_AREG(U(A), 32_REG(1))
* 3 CALL GET_AREG(U(B), 32_REG(2))
* 4 CALL SET_CD(32_REG(1), 32_REG(2))
* 5 CALL HALFWORD_TOGGLE
* 6 RETURN
*
*****
```

NSWC

ANZUYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 134

_MCL --COMPARE LIMITS, REGISTER FMT IV A, F-74 5

REF
PAGE
2345
2347
2348
2349
2350
2351
2352

```
*****
* 1  .. A(UCA)+1 > A(U8)} >= A(U{A}) -> SET ASP(C,1} "OUTSIDE/WITHIN LIMITS.
* 2  CALL GET_AREG(U{A}, 32_REG(2))
* 3  CALL GET_AREG(U{A}+1, 32_REG(3))
* 4  CALL GET_AREG(U{8}, 32_REG(1))
* 5  CALL SET_CD2(32_REG(1), 32_REG(2), 32_REG(3))
* 6  CALL HALFORD_TOGGLE
* 7  RETURN
*****
```

AN/UYK-7 (CP)
CP INSTRUCTION SET

NSWC

_HCR --COMPARE MASHED, REGISTER FMT IV A, F-74 6

```

REF
PAGE *****
* 1 .. A(U(A)+1) .A. A(U(A)) IS COMPARED TO A(U(B)) ; SET ASR(2,2) "CD" ACCORDINGLY.
* 2 CALL GET_AREG(U(A), 32_REG(1))
* 3 CALL GET_AREG(U(A)+1, 32_REG(2))
* 4 32_REG(1) := 32_REG(1) .AND. 32_REG(2)
* 5 CALL GET_AREG(U(B), 32_REG(2))
* 6 CALL SET_CD(32_REG(1), 32_REG(2))
* 7 CALL HALFWORD_TOGGLE
* 8 RETURN
*
*****

```

NSWC

AN/UYK-7 (CP;
CP INSTRUCTION SET

14 DEC 79 PAGE 136

_HCB --COMPARE BU(B) WITH BU(A) FMT IV A F-74 7

REF	PAGE	CODE
2364	*	1 .. BU(B)(15,16) IS COMPARED TO BU(A)(15,16) ; SET ASR(2,2) "CD" ACCORDINGLY
2366	34 *	2 CALL GET_BREG(U(B), 32_REG(1))
2367	34 *	3 CALL GET_BREG(U(A), 32_REG(2))
2368	* 4	32_REG(1) := 32_REG(1)(15)
2369	* 5	32_REG(2) := 32_REG(2)(15)
2370	29 *	6 CALL SET_CD(32_REG(1), 32_REG(2))
2371	* 7	CALL HALFWORD_TOGGLE
2372	* 8	RETURN

MSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_HSM --STORE ICC MONITOR CLOCK IN A PPT IV A P-77 0

2374	REF	*****	
2375	PAGE	*****	
2376		*****	
2377		*****	
2378		*****	
2379		*****	
2380		*****	
2381		*****	
2382		*****	
2383		*****	

1	IF ASP(15,4)=0 "TASK MODE", THEN --PRIVILEGED INSTRUCTION.
2	CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-71, PRIVILEGED INSTRUCTION VIOLATION")
3	--ABORT THE INSTRUCTION.
4	ENDIE
5	CALL CP/ICC_CLOCK_COMMUNICATIONS
6	SET U_BUS INTO 32_REG12
7	CALL PUT_AREGIO(8), 32_REG(2)
8	CALL HALFWORD_TOGGLE
9	REIDEN

NSWC

AP/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 138

_HSTC --STORE REAL-TIME CLOCK IN A FMT IV A F-77 I

REF
PAGE

```
2385 * 1 CALL CP/IOC_CLOCK_COMMUNICATIONS
2386 * 2 GET 0_BUS INTO 32_REG(2)
2387 * 3 CALL PUT_AREG(2), 32_REG(2)
2388 * 4 CALL HALFWORD_TOGGLE
2389 * 5 RETURN
```

NSWC AN/UYK-7 (CPI
C2 INSTRUCTION SET

_MPI(E_HAI) ..PREVENT(ALLOW) CLASS III INT. FMT IV A, F-77 & E F-77 5

REF
PAGE

```

2391 * 1 IE ASRC(19,4)=0 "TASK MODE", THEN ..PRIVILEGED INSTRUCTION.
2392 * 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-1, PRIVILEGED INSTRUCTION VIOLATION)
2393 * 3 ..ABORT THE INSTRUCTION.
2394 * 4 ENDBE
2395 * 5 IE ((F4) = 4 "MPI", T=EN
2396 * 6 ASRC(12,1) := 1 ..LOCKOUT CLASS III INTERRUPTS.
2397 * 7 ELSE .."HAI".
2398 * 8 ASRC(12,1) := 0 ..ENABLE CLASS III INTERRUPTS.
2399 * 9 ENDBE
2400 * 10 CALL HALFWORD_TOGGLE
2401 * 11 RETURN
2402 *

```

NSWC

AN/UYK-7 (CP)
CP INSTRUCTION SET

14 DEC 79 PAGE 140

-H776 --HALT AND WAIT FOR INTERRUPT FMT IV A F-77 6

REF
PAGE

```

2404 * 1
2405 * 2
2407 * 3
2408 * 4
2409 * 5
2410 * 6
2411 * 7
2412 * 8
2413 * 9
2414 * 10
2415 * 11
2416 * 12
2417 * 13

```

```

*****
1 IE ASPL(1,4)=0 "TASK MODE", THEN --PRIVILEGED INSTRUCTION.
  CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-1", PRIVILEGED_INSTRUCTION_VIOLATION)
  --ABORT THE INSTRUCTION.
ENDIE
2 IE "NOT. U(1)", THEN --HALT INSTRUCTION.
  STOP_PROCESSING_INSTRUCTIONS_WITH_STOP_6_INDICATOR_SEY
  --HALT AND WAIT FOR INTERRUPT.
  STOP_PROCESSING_INSTRUCTIONS
  WAIT_FOR_AN_ASYNCHRONOUS_OR_CP_MONITOR_CLOCK_INTERRUPT
  UPON_INTERRUPT_PROCESS_IT_AND_CONTINUE_WITH_NEXT_INSTRUCTION
ENDIE
3 CALL HALFWORD_TOGGLE
  BEIJEN
*****

```

NSWC

AN/UYK-7 (CP)

14 DEC 79 PAGE 141

* AN/UYK-7 CP EMULATION GLOSSARY *
*

NSWC AN/UYK-7 (CP)
AN/UYK-7 CP EMULATION GLOSSARY

AN/UYK-7 CP EMULATION STATUS/CONTROL INDICATORS

```

*****
+ 1 TO SUPPORT REEMPANCY ALL GLOBAL CONTROL VARIABLES ARE CONSIDERED
+ 2 TO BE ACCESSIBLE BY ALL PROCEDURES BUT CONTAINED WITHIN A WORK SPACE
+ 3 FOR A PARTICULAR EMULATOR.
+ 4
+ 5 CHARACTER_ADDRESSING_OVERRIDE - WHEN SET, CHARACTER ADDRESSING IS
+ 6 IN EFFECT. USED TO ACQUIRE CORRECT
+ 7 CHARACTER SPECIFICATION VALUES I.E.
+ 8 -C, -P, AND -MASK WHEN REPEATING
+ 9 INSTRUCTIONS USING CHARACTER ADDRESSING.
+ 10 CP_MONITOR_CLOCK_INTERRUPT - WHEN SET, THE CP MONITOR CLOCK HAS GONE
+ 11 NEGATIVE AND THE ASSOCIATED CLASS II
+ 12 INTERRUPT IS HELD PENDING UNTIL PROCESSED.
+ 13 EXECUTE_REMOTE_IN_PROGRESS - WHEN SET, AN INSTRUCTION IS BEING EXECUTED
+ 14 VIA THE XR (F=02 2) OR XRL (F=02 3)
+ 15 INSTRUCTION. USED TO SUPPRESS CHANGING OF
+ 16 THE ASR UPPER/LOWER HALF-WORD BIT WHEN A
+ 17 HALF-WORD INSTRUCTION IS EXECUTED VIA THE
+ 18 XR OR XRL INSTRUCTION.
+ 19 INSTRUCTION_FORMAT_INDICATOR - THIS INDICATOR CONTAINS THE FORMAT
+ 20 (I.E. I, II, III, IV) OF THE INSTRUCTION
+ 21 THAT IS CURRENTLY EXECUTING.
+ 22 INTERPROCESSOR_INTERRUPT - WHEN SET, AN INTERPROCESSOR INTERRUPT IS
+ 23 PENDING. THIS DATA STRUCTURE MUST RECEIVE
+ 24 THE CP ID OF THE PROCESSOR CAUSING THE
+ 25 INTERRUPT.
+ 26 INTERRUPT_SCAN_INHIBIT - WHEN SET (I.E. BETWEEN EXECUTION OF
+ 27 TWO HALF-WORD INSTRUCTIONS AND DURING
+ 28 REPEATS OF CBR REFERENCE INSTRUCTIONS
+ 29 (F=54-57)), THE ASYNCHRONOUS INTERRUPT
+ 30 SCAN IS NOT PERFORMED. FURTHERMORE, THE
+ 31 MAIN LOOP LOGIC SUPPRESSES INTERRUPT
+ 32 SCANNING WHEN IN INTERRUPT MODE.
+
*****

```

STATUS/CONTROL INDICATORS PAGE 2

2453 * 1 MEMORY_STORE_INDICATOR - THIS INDICATOR IS SET WHEN AN INSTRUCTION
2454 * 2 STORES A QUANTITY INTO MEMORY. THIS FACT,
2455 * 3 IS REQUIRED FOR REPEAT TERMINATION LOGIC.
2456 * 4 PSEUDO_BREAKPOINTS - WHEN SET, ALL BREAKPOINT REGISTERS WILL
2457 * 5 BE ACTIVE (TOTAL OF 8), ELSE ONLY THE
2458 * 6 EMULATED HARDWARE REGISTER WILL BE ACTIVE.
2459 * 7 NOTE, THE UYK-7 EMULATOR HAS EIGHT (8)
2460 * 8 BREAKPOINT REGISTERS. ONE REGISTER
2461 * 9 CORRESPONDS TO THE ACTUAL HARDWARE
2462 * 10 BREAKPOINT REGISTER OF A REAL UYK-7 MACHINE.
2463 * 11 THE OTHER SEVEN (7) BREAKPOINT REGISTERS
2464 * 12 ARE CALLED PSEUDO BREAKPOINT REGISTERS AND
2465 * 13 ARE CONSIDERED EXTENSIONS TO THE UYK-7
2466 * 14 ARCHITECTURE. PSEUDO BREAKPOINTS ARE NOT
2467 * 15 ACCESSIBLE TO UYK-7 PROGRAMS.
2468 * 16 REPEAT_IN_PROGRESS - WHEN SET, INSTRUCTION IS BEING REPEATED.
2469 * 17 REPEAT_PENDING - SET BY EXECUTION OF A REPEAT INSTRUCTION.
2470 * 18 REPEAT IS NOT IN PROGRESS UNTIL THE REPEATED
2471 * 19 INSTRUCTION HAS BEEN FETCHED FROM MEMORY.
2472 * 20 SPR_CHECKS - WHEN SET, SOFTWARE PROTECTION REGISTER
2473 * 21 CHECKS ARE ENABLED. SPR CHECKS ARE
2474 * 22 DISABLED WHEN CLASS II INTERRUPTS ARE
2475 * 23 LOCKED OUT, INTERRUPT BASE REGISTERS ARE
2476 * 24 SELECTED OR MEMORY LOCKOUT INHIBIT IS SET.
2477 * 25 SPR_PRIVILEGED_INSTRUCTION - WHEN SET, ONE OF THE CONDITIONS CAUSING
2478 * 26 USE OF INDIRECT ADDRESSING WITH SPR(16,1)
2479 * 27 TO BE A PRIVILEGED OPERATION IS IN EFFECT.
2480 * 28 THESE CONDITIONS INCLUDE:
2481 * 29 (1) CERTAIN DOUBLE-WORD INSTRUCTIONS
2482 * 30 (ALL EXCEPT OS 4).
2483 * 31 (2) ALL REPEATED INSTRUCTIONS.
2484 * 32 (3) U(F) = 25 OR U(F) = 11.
2485 * 33 (4) ALL FORHAT III INSTRUCTIONS.

AN/UYK-7 (CP)
AN/UYK-7 CP EMULATION GLOSSARY

NSMC

AN/UYK-7 CP ARCHITECTURE DESCRIPTORS

2487	* 1	BREAKPOINT_REGISTER	- CME ADDRESS OF UYK-7 HARDWARE BREAKPOINT REGISTER (60).
2488	* 2		- (CONTROL MEMORY REGISTERS) SECTION OF CP WHERE THE ADDRESSABLE REGISTERS ARE LOCATED.
2489	* 3	_CME	INCLUDED APE TASK AND INTERRUPT ACCUMULATORS, INDEX(1), AND BASE(S) REGISTERS. SMP'S AND SMP'S, THE AS9, CP MONITOR CLOCK AND BREAKPOINT REGISTER(A) AND ICM AND USM FOR EACH INTERRUPT LEVEL. CME LOCATION 10 IS UNASSIGNED BIT ADDRESSABLE BY PROGRAMS.
2490	* 4		- MAIN MEMORY SUPPLYING 32-BIT WORD STORAGE.
2491	* 5		- AMOUNT OF 32-BIT MAIN MEMORY (UP TO 256 K) AVAILABLE FOR USE BY CP
2492	* 6		- 512 WORD NON-DESTRUCTIVE READ OUT MEMORY. USUALLY CONTAINS DIAGNOSTIC, BOOTSTRAP, RECOVERY AND INTERRUPT SWITCHER PROGRAMS.
2493	* 7		
2494	* 8		
2495	* 9		
2496	* 10		
2497	* 11		
2498	* 12	_MAIN	
2499	* 13		
2500	* 14	MEMORY_LIMIT	
2501	* 15		
2502	* 16	_NORC	
2503	* 17		
2504	* 18		
2505	* 19		

NS=C

AN/UYK-7 (CP)
AN/UYK-7 CP EMULATION GLOSSARY

14 DEC 79 PAGE 145

AN/UYK-7 CP MAINTENANCE PANEL SUPPORT

```

*****
*
* 1  BCDSTRAP_SWITCH      - A 3-POSITION SWITCH (0,1,2) WHICH
* 2                        SELECTS AN MDRPC STARTING ADDRESS.
* 3  _STEP                - WHEN SET, THE EMULATOR SUSPENDS OPERATION
* 4                        AT THE BEGINNING OF THE MAIN LOOP UNTIL
* 5                        RESTARTED. THIS CORRESPONDS TO THE MODE
* 6                        SWITCH (I.E. SET -> INSTRUCTION MODE
* 7                        AND CLEAR -> RUN MODE).
* 8  STOP_SWITCH          - SERIES OF THREE SWITCHES (NUMBERED
* 9                        5-6-7) FOUND ON THE MAINTENANCE PANEL
* 10                       WHICH ALLOW STOPPING UNDER PROGRAM CONTROL.
* 11  JUMP_SWITCH         - SERIES OF THREE SWITCHES (NUMBERED
* 12                       1-2-3) FOUND ON THE MAINTENANCE PANEL
* 13                       WHICH ALLOW BRANCHING UNDER PROGRAM CONTROL.
*
*****

```

AN/UYK-7 CP EMULATION INTERNAL REGISTER SUPPORT

2521	1	-C	THE C DESIGNATOR FOR CHARACTER ADDRESSING
2522	2		C FIELD OF LAST EXECUTED IAW.
2523	3	W	THE MEMORY REFERENCE REGISTER. A 32 BIT
2524	4		REGISTER CONTAINING THE ADDRESS OF THE LAST
2525	5		MEMORY LOCATION REFERENCED.
2526	6	-MASK	THE CHARACTER MASK NEEDED TO ACQUIRE
2527	7		bits FOR CHARACTER ADDRESSING.
2528	8	P	THE PROGRAM COUNTER REGISTER. A 20-BIT
2529	9		REGISTER WITH LOW ORDER 16 BITS (P10))
2530	10		FORMING A DISPLACEMENT THAT IS ADDED TO THE
2531	11		BASE REGISTER INDICATED BY BITS 19 THROUGH
2532	12		17.
2533	13		BIT 16 IS UNUSED.
2534	14	-P	LEAST SIGNIFICANT BIT ADDRESS FOR
2535	15		CHARACTER ADDRESSING.
2536	16	REPEAT_AB	CONTAINS THE A & B FIELDS OF THE LAST
2537	17		REPEAT INSTRUCTION AS USED IN REPEAT
2538	18		TERMINATION LOGIC.
2539	19	REPEAT_ACCUMULATOR	A 32-BIT WORKING REGISTER WHICH HOLDS
2540	20		A COPY OF THE APPROPRIATE ACCUMULATOR TO
2541	21		TESTED BY REPEAT TERMINATION LOGIC. IT IS
2542	22		THE RESPONSIBILITY OF EACH INSTRUCTION
2543	23		(WHERE APPLICABLE) TO ENSURE THE PROPER
2544	24		CONTENTS OF THIS REGISTER.
2545	25	REPEAT_SY	CONTAINS THE SY MODIFIER OF THE LAST
2546	26		REPEAT INSTRUCTION AS USED IN REPEAT
2547	27		TERMINATION LOGIC.
2548	28	U	THE INSTRUCTION REGISTER. A 32-BIT
2549	29		REGISTER WHICH RECEIVES THE INSTRUCTION WORD
2550	30		FROM MEMORY AND PROVIDES THE CONTROL
2551	31		NECESSARY TO START EXECUTION OF THE
2552	32		INSTRUCTION. FIELDS (E-G, F, F2, A, ETC.)
2553	33		ARE AS PER REPERTOIRE CARD.
2554	34	UL	LOWER 16 BITS OF U.
2555	35	UU	UPPER 16 BITS OF U.
2556	36		
2557	37		NOTE: 32_REG IS A GENERIC TERM FOR A 32-BIT REGISTER. THIS DESIGN
2558	38		ASSUMES 4 SUCH WORKING REGISTERS WHICH ARE REFERENCED AS
2559	39		32_REG(1),...,32_REG(4). THESE REGISTERS DO NOT NECESSARILY
2560	40		CORRESPOND TO ANY SPECIFIC UYK-7 INTERNAL REGISTERS.
2561	41		REFERENCE PARAMETERS ARE INDICATED BY THE 'REF'
2562	42		CONSTRUCT IN THE SUBPROCEDURE DECLARATION. CHANGING A
2563	43		REFERENCE PARAMETER MODIFIES THE ARGUMENT IN THE CALLING
2564	44		ROUTINE.

NSWC

AN/UYK-7 (CP)

14 DEC 79 PAGE 147

*
* REFERENCES *
*

NSWC

AN/UYNK-7 (CP)
REFERENCES

14 DEC 79 PAGE 148

REFERENCES

2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696

1. CAINE, FARBER, AND GORDON, "PROGRAM DESIGN
LANGUAGE REFERENCE GUIDE", JULY 1975.
2. NAVSHIPS 0967-051-6291, AN/UYNK-7(V) COMPUTER SET,
MARCH 1977.
3. SPERRY UNIVAC, STUDY GUIDE FOR AN/UYNK-7 COMPUTER
MAINTENANCE COURSE VOLUMES 1, 2 & 3, JANUARY 1977.
4. NAVSHIPS 0967-319-4012, TECHNICAL MANUAL FOR
COMPUTER SET AN/UYNK-7(V) VOLUMES 1 & 2, JANUARY 1974.
5. SPERRY UNIVAC, STUDY GUIDE FOR AN/UYNK-7 COMPUTER,
VOLUME 1, MARCH 1973.

NSWC

AN/UUK-7 (CP)

14 DEC 79 PAGE 149

*
* INDEX TO DATA ITEMS *
*

INDEX TO DATA ITEMS

<u>PAGE</u>	<u>LINE</u>	<u>TYPE</u>	<u>NAME AND REFERENCES</u>
DI			ABS_ADDR 17 BPR_CHECK 11 17 20
DI			ACCUMULATOR_SIGN 96 DO_MULTIPLY 25 29
DI			ALTO_RECOVERY 8 INTERRUPT_SEQUENCE 19
DI			A_DESIGNATOR 28 UPDATE_REPLACE 1 4 6
	33		GET_AREG 1 2 6 6 8 10
	36		PUT_AREG 1 2 5 5 7 9
DI			A_MANTISSA 40 FLOATING_ADD_SUBTRACT_HEADER 2 9 10 14 23
DI			BCOTSTRAP_SWITCH 8 INTERRUPT_SEQUENCE 21
DI			BREAKPOINT_REGISTER 17 BPR_CHECK 25 26 27 28
DI			B_DESIGNATOR 34 GET_BREG 2 6 6 7 11 23
	37		PUT_BREG 2 6 6 7 9 11
DI			CALL_REPLACE 92 _8Z 9
DI			CHARACTER_ADDRESSING_OVERRIDE 4 I_SEQUENCE 17

INDEX TO DATA ITEMS

<u>PAGE</u>	<u>LINE</u>	<u>TYPE</u>	<u>NAME AND REFERENCES</u>
		DI	CHARACTER_ADDRESSING_OVERRIDE 14 DP_READ 12 15 NP_STORE 12 18 IA_SEQUENCE 38
		DI	CHECK_TYPE 20 SPR_CHECK 9 14 15
		DI	CP_MONITOR_CLOCK_INTERRUPT 5 INTERRUPT_SCAN 10 13
		DI	DUP_C 13 JUMP_ADDRESS 7
		DI	DUM_CI 13 JUMP_ADDRESS 7
		DI	DUM_MASK 13 JUMP_ADDRESS 7
		DI	DUM_P 13 JUMP_ADDRESS 7
		DI	EXECUTE_REMOTE_IN_PROGRESS 4 I_SEQUENCE 16 25 HALF-WORD_TOGGLE 5 55 _NR 17
		DI	FLOATING_ADD_SUP_HDR 65 _PA 1 66 _FAN 1

MSMC

AMFJYK-7 (CP)
INDEX TO DATA ITEMS

14 DEC 79 PAGE 149.003

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
		DI	FLOATING_POINT_END 68 _FD 48
		DI	GENERATE_SYNCHRONOUS_INTERRUPT 119 HSC_60 6 120 HLC_61 6
		DI	HALFWORD_TOGGLE 119 HSC_60 9 120 HLC_61 9 121 _MLC 16 122 _MDLC 12 123 _MSF 15 124 _MDSF 16 125 _MCP 4 127 _MCR 18 128 _MH 12 129 _MD 8 130 _MPT 18 132 _MLB 6 133 _MC 5 134 _MCL 6 135 _MCM 7 136 _MCB 7 137 _MSIM 6 138 _MSTC 4

NSWC AN/UYK-7 (CPI)
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
	139		_HPI
	10		
	140		_H77b
	12		
DI			INSTRUCTION_FORMAT_INDICATOR
	4		I_SEQUENCE
	20		
DI			INSTRUCTION_FORMAT_INDICATOR
	14		OP_HEAD
	18		
	30		
	15		OP_STORE
	20		
	26		
	55		_XF
	18		
DI			INTERPROCESSOR_INTERRUPT
	5		INTERUPT_SCAN
	19		
	22		
DI			INTERUPT_SCAN_INTERRUPT
	3		CP MAIN LOOP
	17		
	25		
	28		
	4		I_SEQUENCE
	28		
	9		REPEAT_SEQUENCE
	32		
DI			ICC_
	12		GET_ISC
	6		
	7		
	10		
	12		
DI			ICC_CLOCK_COMMUNICATIONS
	137		_MSIM
	5		
	138		_MSYC
	1		
DI			JUMP_STOP
	115		_J532
	3		
	4		
	9		
	28		
	116		J533
	4		
	5		
	10		
	22		
	24		
DI			JUMP_SWITCH
	115		_J532
	14		

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			116 J533 15
DI			MAIN_MEMORY 19 MEMORY_READ 9 14
DI			MB_CARRY 96 DO_MULTIPLY 4 8 9 15 16 24 43 47
DI			MEMORY_STORE_INDICATOR 3 CP MAIN LOOP 13 9 REPEAT_SEQUENCE 25 26 15 OP_STORE 19
DI			M_DISPLACEMENT 13 JUMP_ADDRESS 3 9 11 14 OP_READ 4 7 10 10 15 OP_STORE 4 7 10 10
DI			M_MANTISSA 40 FLOATING_ADD_SUBTRACT_HEADER 1 6 7 11 19
DI			OPERAND_DISPLACEMENT 13 JUMP_ADDRESS 5 7 9 9 11 14 15 14 OP_READ 7 7 15 OP_STORE 7 7 55 _XR 3 6 6 105 _JEP 2 106 _DJZ 2 7 107 _DJNZ 2 7 108 _FSI 2 13

NSWC

AN/UUK-7 (CP)
INDEX TO DATA ITEMS

14 DEC 79 PAGE 149.906

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
	109		_LBJ 8 10
	110		_JBNZ 6 8
	112		_JL 4 7
	113		_J530 2 7
	114		_J531 3 32
	115		_J532 24 27
	116		J533 2 23
DI	OPERAND_5		
	13		JUMP_ADDRESS 12 15 16 17
	55		_XR 3 7
	105		_JEP 2 16
	106		_JLZ 2 7
	107		_DJNZ 2 7
	108		_F51 2 13
	109		_LBJ 8 10
	110		_JBNZ 6 8
	112		_JL 4 6
	113		_J530 2 7
	114		_J531 3 32
	115		_J532 24 27
	116		J533 2 23
DI	OPERAND_DISPLACEMENT		
	105		_JEP 16

NSUC AN/UYK-7 (CP)
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
DI	12	O_BUS	GET_ISC 8 10
			CP/IOC_CLOCK_COMMUNICATIONS 8 9 10 11 12 13
			_AEI 9 11 11 12 13
			_LIM 9 11 11 12 13
			_IO 14 15 15
			_MSIM 6
			_MSTC 2
DI	17	PSEUDO_BREAKPOINTS	BPR_CHECK 7
DI	17	PSEUDO_BREAKPOINTS	BPR_CHECK 19
DI	19	P_MOD	MEMORY_READ 8 11
DI	11	P_MODIFICATION	GENERATE_SYNCHRONOUS_INTERRUPT 4 44
			SPR_CHECK 10 18 24 30 34
DI	9	REPEAT_AB	REPEAT_SEQUENCE 8 19
			_REPLACE 5
			_RP 13
DI	9	REPEAT_ACCUMULATOR	REPEAT_SEQUENCE 20 21 22 23 25 26
			_OP 2 3 3 4

MSWC

AN/U R-7 (CP)
INDEX TO DATA ITEMS

14 DEC 79 PAGE 149.008

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
49	SC	3	4 4 5
50	MS	3	5 5 6
51	XOR	2	3 3 4
52	ALP	1	4 4 6
53	LLP	3	5 5 8 8 12 14
54	CNT	2	6 6 10
56	SLP	1	3 3 4
57	SSUM	1	3 3 9 10
58	SDIF	2	3 3 9 10
60	TSF	4	5 10 11
67	FM	14	15 16 17 21 24 26
76	LA	1	2
77	LXB	2	3
78	LDIF	1	3 3 5
79	ANA	2	3 3 5
80	AA	2	3 3 5
81	LSUM	2	3 3 5
82	LNA	1	2 2 3
83	LM	1	2 3 3 5
85	AB	4	6 7
86	ANB	3	4 6 7
87	SB	2	3 3 4
88	SA	2	3
90	SNA	2	3 3

NSWC AN/UYM-7 (CP)
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
91	1	SM	1 2 3 3 5
92	2	BZ	3 5 7 9
93	3	RA	1 4 4 6 6 9
94	4	PI	1 3 3 5 5 8
95	5	M	7 7 9
96	6	DD_MULTIPLY	1 4 9 9 12 13 13 16 16 19 19 21 22 22
			25 26 26 27 28 28 30 31 33 36 37 37 39 44
			45 45 48 48 49 50 50
117	LCT		11 12
118	SCY		10 10 12
128	HM		7 7 10
DI	REPEAT_IN_PROGRESS		
	3	CP MAIN LOOP	12 21
	4	I_SEQUENCE	23
	5	INTERUPT_SCAN	28 33
	9	REPEAT_SEQUENCE	5 9 10 11 12 13 14 15 16 20 21 22 23 25
	11	GENERATE_SYNCHRONOUS_INTERRUPT	26 31
	18	IA_SEQUENCE	42
	27	_REPLACE	29
	117	LCT	5
	118	SCY	6 13
DI	REPEAT_PENDING		
	4	I_SEQUENCE	21 24 32
	5	INTERUPT_SCAN	33
	11	GENERATE_SYNCHRONOUS_INTERRUPT	41

NSWC AN/UYK-7 (CP)
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			75 _RP 12
DI			REPEAT_SY 9 REPEAT_SEQUENCE 3
			75 _RP 14
DI			RESUME_TYPE 19 MEMORY_READ 5 11
DI			ROUND_BIT 68 _FD 31 33
DI			SHIFT64_REG 122 _HDLC 9
DI			SHIFT_COUNT 24 GET_SHIFT_AMOUNT 3 5 9 12
			40 FLOATING_ACD_SUBTRACT_HEADER 17 16 19 21 22 23
			42 FLOATING_NORMALIZE 4 6 8 9 10
			121 _HLC 2 5 6 7
			122 _HDLC 2 6 7 8
			123 _HSF 3 5 7 8 9 13
			124 _HDSF 4 6 8 9 14
DI			SIGN_DIVIDEND 97 DO_DIVIDE 10 26
DI			SIGN_IND 68 _FD 6 43
			97 DO_DIVIDE 5 23

NSWC ANALYK-7 (CP)
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
DI			SIGN_INDICATOR
	67	FM	6 23
	95	M	4 6
	128	HM	4 6
DI			SPR_CHECKS
	3	CP MAIN LOOP	8 10
	20	SPR_CHECK	12
DI			SPR_PRIVILEGED_INSTRUCTION
	4	I_SEQUENCE	18 25
	20	SPR_CHECK	16
	40	FLOATING_ADD_SUBTRACT_HEADER	4
	59	DS	1
	61	DL	1
	62	DA	2
	63	DC	3
	67	FM	1
	68	FD	1
	77	LXB	1
	89	SXB	1
	105	JEP	2
	106	DJZ	1
	107	DJNZ	1
	109	F51	1
	109	LEJ	1

NSWC AN/UYK-7 (CP)
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
	110		_JBNZ 1
	112		_JL 1
	113		_J530 1
	114		_J531 2
	115		_J532 1
	116		J533 1
DI			STEP_SWITCH
	115		_J532 10
	116		J533 11
DI			S_DESIGNATOR
	14		OP_READ 6 11 19 20 22
	15		OP_STORE 6 11 21 22 24
	18		IA_SEQUENCE 7 9 11 13 18 20 22 24
	19		MEMORY_READ 4 7
	23		DC_JUMP 3 5
	32		ADD_S 2 7 8
	35		GET_SREG 1 2 5 5 7 9
	38		PUT_SREG 1 2 4 4 6 8
	73		_IC 6 8 12
DI			S_INDICATOR
	20		SDE_CHECK 6 13
DI			TYPE_
	17		SDE_CHECK 4 18 26 31 38

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
		DI	UPDATE_REPLACE 93 _RA 9 94 _RI 8
		DI	16_REG 20 SPR_CHECK 11 29
		DI	18_REG 19 MEMORY_READ 2 7 8 10 11 14
		DI	32_REG 4 I_SEQUENCE 3 6 6 7 9 11 12 19 13 JUMP_ADDRESS 14 14 16 17 18 14 OP_READ 3 22 24 26 26 31 34 35 15 OP_STORE 3 27 18 IA_SEQUENCE 10 11 12 13 19 MEMORY_READ 3 8 14 24 GET_SHIFT_AMOUNT 8 9 11 12 26 REPLACE_CHECK 3 5 27 _REPLACE 1 4 8 8 11 11 28 UPDATE_REPLACE 3 6 7 32 ADD_5 8 9 33 GET_AREG 1 4 8 10 34 GET_BREG 1 4 8 11 13 35 GET_SREG 1 3 7 9 36 PUT_AREG 1 4 7 9 37 PUT_BREG 1 4 9 11

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
38	PUT_SREG		
	1	3	6 8
40	FLOATING_ADD_SUBTRACT_HEADER		
	5	8 12 12 15	15 17 17 25
41	FLOATING_OVERFLOW		
	3	6 6	
42	FLOATING_NORMALIZE		
	2	10 10 12	
43	FLOATING_ROUND		
	2	10	
45	DIVIDE_COMPARE		
	2	3 4	
48	_CR	1	3
49	_SC	1	2 2 4
50	_MS	1	2 4 4 5 5
51	_XOR	1	3
52	_ALP	2	3 4 4
53	_LLP	4	5 7 8
54	_CNT	1	5
55	_XR	6	7 8 12
56	_SLP	2	3
57	_SSUM	2	3
58	_SDIF	1	3
59	_DS	2	3 4 5
61	_DL	2	3 4 5
64	_LBMP	1	2 3 4 5 14 15 17
65	_FA	3	5 7 10 12
66	_FAN	3	5 7 10 12
67	_FM	2	3 4 5 6 7 6 7 6 9 10 13 13 14
		14 14 15 17 18 18 21 26	
68	_FD	2	3 4 4 6 7 9 11 14 14 14 15 19 27 32

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES	34	37	39	46	12	16	19	21	21
	69	_XS		4	5							
	70	_IPI		8	9	9	10	11	15			
	71	_AEI		8	9	9						
	72	_LIM		8	9	9						
	73	_IO		8	9	9	11					
	77	_LXB		9	10							
	78	_LDIF		4	5	5	6					
	79	_ANA		1	3							
	80	_AA		1	3							
	81	_LSUM		1	3							
	84	_LB		2	3	4	4	5				
	85	_AB		2	3	4	5	5	6			
	86	_ANR		2	4	5	5	5	6			
	89	_SKB		4	5	5	6					
	92	_BZ		1	5	7	10					
	93	_RA		2	4	6						
	95	_M		2	3	4	4	7	7	10		
	96	DO_MULTIPLY		1	1	1	3	3	6	9	12	12
				21	26	27	27	27	33	36	36	36
				49	49							
	97	DO_DIVIDE		1	5	6	7	9	11	15		
	98	_D		2	5							
	99	_BC		3	4							
	100	_CAI		1	2	3	3	5	8	8	10	
	101	_C		2	3	4						

INDEX TC DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
	131	OC_SORT	3 3 3 4 5 5
01	38	PUI_SREG	2 3
	40	FLOATING_ADD_SUBTRACT_HEADER	2
01	108	_CASE	108 _F51 6
	127	_HGR	3
01	5	_CLASS	5 INTERRUPT_SCAN 7 8 12 16 17 21 34
	8	INTERRUPT_SEQUENCE	4 6 7 8 9
	12	GET_ISC	4 8 11
	74	_IP	5 6 7
01	5	_CODE	5 INTERRUPT_SCAN 5 8 11 17 20 27 34
	11	GENERATE_SYNCHRONOUS_INTERRUPT	6 24 25 26
	12	GET_ISC	5 12
01	14	_CI	14 OP_READ 6 25
	15	OP_STORE	6
	19	IA_SEQUENCE	28
02	5	_DFCODE	5 CP MAIN LGDP 20
	55	_AR	19

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
		DI	_DISPLACEMENT 23 DO_JUMP 4 6
		DI	_ENDCASE 127 _HOR 16
		DI	_INTERRUPT 11 GENERATE_SYNCHRONOUS_INTERRUPT 5 7 13 23
		DI	_ISC 8 INTERRUPT_SEQUENCE 5 7 19
		DI	_JUMP 114 _J531 5 10 14 19 25 28 31
		DI	_MASK 14 OP_READ 6 13 35 15 OP_STORE 6 13 18 IA_SEQUENCE 4 37
		DI	_PATCH 115 _J532 2 11 15 19 26 28 116 J533 3 12 16 20 22 24
		DI	_NDRO 19 MEMORY READ 8
		DI	_NLP 53 _LLP 6
		DI	_P 14 OP_READ 6 13 35 15 OP_STORE 6 13

NSWC AN/JYK-7 (CP)
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES	
			18	IA_SEQUENCE 4 36

NSWC

AN/UUK-7 (CP)

14 DEC 79 PAGE 150

*
* INDEX TO FLOW SEGMENTS *
*

NSWC AN/UYK-7 (CPI)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
32	FS	ADD_S	4 I_SEQUENCE 7 13 JUMP_ADDRESS 16 14 OP_READ 20 15 OP_STORE 22 18 IA_SEQUENCE 11 55 _XR 7 73 _IO 12 111 _JS 5 112 _JL 10
17	FS	BPR_CHECK	4 I_SEQUENCE 9 13 JUMP_ADDRESS 18 14 OP_READ 21 15 OP_STORE 23 18 IA_SEQUENCE 12 55 _XR 8 111 _JS 6 112 _JL 11
104	FS	CG	
22	FS	CP/IOC_CLOCK_COMMUNICATIONS	137 _HSIM 5 138 _HSTC 1

NSWC AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
3		FS	CP MAIN LOOP
62		FS	OA
45		FS	DIVIDE_COMPARE 68 _FD 15 19
			97 00_DIVIDE 15
97		FS	00_DIVIDE 98 _D 5
			129 _HD 5
23		FS	00_JUMP 105 _JEP 16
			106 _DJZ 7
			107 _DJNZ 7
			108 _F51 13
			109 _LBJ 10
			110 _JBNZ 8
			113 _J530 7
			114 _J531 32
			115 _J532 27
			116 J533 23
96		FS	00_MULTIPLY 67 _FM 14
			95 _M 5
			128 _HM 5

NSWC AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
131	FS	DC_SORT	13C _MRT 11 14
40	FS	FLOATING_ADD_SUBTRACT_HEADER	
42	FS	FLOATING_NORMALIZE	65 _FA 7
			66 _FAN 7
41	FS	FLOATING_OVERFLOW	43 FLOATING_ROUND
			65 _FA 10
			66 _FAN 5
			68 _FD 27 32
46	FS	FLOATING_POINT_END	65 _FA 12
			66 _FAN 12
			67 _FM 26
43	FS	FLOATING_ROUND	65 _FA 10
			66 _FAN 10
			67 _FM 21
11	FS	GENERATE_SYNCHRONOUS_INTERRUPT	17 BPF_CHECK 39 42
			18 IA_SEQUENCE 30 32
			19 MEMORY_READ 11
			20 SPP_CHECK 18 24 30 34

NSWC AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
	46		FLUATING_POINT_END 10
	60	TSF	2
	64	LBMP	6 11
	68	FD	12
	69	XS	6
	70	IPI	2
	71	AEI	2 14
	72	LIM	2 14
	73	IO	2 17
	74	IR	2
	112	JL	17
	115	J532	5
	116	J533	6
	117	LCT	7
	118	SCT	7
	137	MSIM	2
	139	HPI	2
	140	M776	2
33	FS	GET_AREG	24
		GET_SHIFT_AMOUNT	11
	40	FLUATING_ADD_SUBTRACT_HEADER	9
	48	GR	2
	49	SC	3
	50	MS	2 3

NSWC

AN/UYK-7 (CPI)
INDEX TO FLOW SEGMENTS

14 DEC 79 PAGE 150.005

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
51		_XOR	2
52		_ALP	2 3
53		_LLP	4 7
56		_SLP	1 2
57		_SSUP	1 2
58		_SDIF	1 2
59		_DS	1 2
62		DA	2 3
63		_DC	5 6
65		_FA	4 5
66		_FAN	3
67		_FM	3
68		_FD	4 5
78		_LDIF	4 5
79		_ANA	2
80		_AA	2
81		_LSUM	2
88		_SA	2
90		_SNA	2
91		_SM	2
93		_RA	1
95		_M	2
98		_O	2
101		_C	3 4
			2

NSWC

AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

14 DEC 79 PAGE 150.006

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
	102	_CL	2 3
	103	_CM	2 3
	104	CG	3 5
	105	_JEP	5 6
	106	_JZ	4 5
	107	_JNZ	4 5
	108	_F51	4
	120	HLC_61	13
	121	_HLC	3
	122	_HDL	3 4
	123	_HSF	1
	124	_HDSF	1 2
	125	_HCP	1
	126	_HDCP	1
	127	_HCR	1 2
	128	_HM	2 3
	129	_HD	2 3 4
	130	_HRT	3 13
	133	_HC	2 3
	134	_HCL	2 3 4
	135	_HCP	2 3 5
24	FS	GET_SPEG	24 GET_SHIFT_AMOUNT
			6
			24 _LEMP
			3

MSWC AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
	69	XS	4
	70	IPI	8
	71	AEI	8
	72	LIM	8
	73	IO	8
	77	LXB	9
	84	LB	4
	85	AB	3
	86	ANB	3
	87	SB	3
	89	SXB	2
	100	CXI	4
	110	JBNZ	1
	111	JS	2
	132	MB	1
	136	HCB	2 3
12		FS	GET_IS 5 INTERRUPT_SCAN 8 17
24		FS	GET_SHIFT_AMOUNT 121 HLC 2
39		FS	GET_SRES 32 ADD 8

NS=C AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
25		FS	HALF-WORD_TOGGLE
120		FS	HLC_61
119		FS	HSC_60
18		FS	IA_SEQUENCE
	13		JUMP_ADDRESS
	14		OP_READ 7
	15		OP_STORE 6
	69		_XS 6
	70		_IPI 2
	71		_AEI 6
	72		_LI 6
	73		_IO 6
	75		_RF 6
			7
5		FS	INTERRUPT_SCAN
	3		CP MAIN LOOP
			18
6		FS	INTERRUPT_SEQUENCE
	5		INTERRUPT_SCAN
			34
	11		GENERATE_SYNCHRONOUS_INTERRUPT
			46
4		FS	I_SEQUENCE
	3		CP MAIN LOOP
			15
13		FS	JUMP_ADDRESS
	55		_XR
			3
	105		_JEP
			2
	106		_DJZ
			2

MSWC

AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

14 DEC 70 PAGE 159.000

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
	107		_DJNZ 2
	108		_FSI 2
	109		_LBJ 8
	110		_JBNZ 6
	112		_JL 4
	113		_J530 2
	114		_J531 3
	115		_J532 24
	116		J533 2
116		FS	J533
117		FS	LCT
19		FS	MEMORY_READ 4
			I_SEQUENCE 12
			JUMP_ADDRESS 17
			OP_READ 22
			OP_STORE 24
			IA_SEQUENCE 13
			_XR 12
			_JL 13
14		FS	OP_READ 40
			FLOATING_ADD_SUBTRACT_HEADER 5 6
			_OP 1
			_SC 1
			_43 1

NSWC

AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

14 DEC 79 PAGE 150.010

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
51		_XOR	1
52		_ALP	1
53		_LLP	3
54		_CNT	1
60		_TSF	4
61		.DL	2 3
62		DA	3 4
63		_DC	6 7
64		_LBP	1 2
67		_FH	2 3
68		_FD	2 3
75		_IA	1
77		_LXB	2
78		_LCIF	1
79		_ANA	1
80		_AA	1
81		_SUM	1
82		_LNA	1
83		_LM	1
84		_LB	2
85		_AB	2
86		_ANB	2
87		_BZ	3
88		_FA	1

NSWC AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
	94		_RI 1
	95		_M 3
	98		_D 2
	100		_CHI 2
	101		_C 3
	102		_CL 4
	103		_CM 4
	104		CG 2
	117		LCT 11
15		FS	DP_STORE 27
			_REPLACE 8 11
	56		_SLP 4
	57		_SSUM 10
	58		_SDIF 10
	59		_DS 4 5
	60		_TSF 11
	67		_SB 4
	68		_SA 3
	90		_SNA 4
	91		_SM 5
	115		_J532 23
	118		SCY 12
36		FS	PUT_AREG 28
			UPDATE_REPLACE 6

NAME AN/UYM-7 (CP)
INDEX TO FLC SEGMENTS

INDEX TO FLC SEGMENTS

LINE	TYPE	NAME AND REFERENCES
40	FLOATING_ADD_SUBTRACT_HEADER	
25		
46	FLOATING_POINT_END	
53	_LLP	12
54	_CNT	10
57	_SSUM	9
58	_SDIF	9
61	_DL	4 5
62	_DA	17 18
67	_FM	9 10
76	_LA	2
77	_LXB	3
78	_LDIF	5
79	_ANA	5
80	_AA	5
81	_LSUM	5
82	_LNA	3
83	_LM	5
95	_M	9 10
98	_D	6 7
119	_HSC_60	27
121	_HLC	9
122	_HOLC	10 11
123	_PSF	10 13
124	_OSF	13 14

NSWC

AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

14 DEC 79 PAGE 150-013

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
			125 _HCP 3
			126 _HDCP 3
			127 _HGR 17
			128 _HM 10 11
			129 _MD 6 7
			130 _HRT 16 17
			137 _HSIM 7
			138 _HSTC 3
37		FS	PUT_BREG 77 _LXB 6
			84 _LB 5
			85 _AB 7
			86 _ANB 7
			89 _SX8 6
			100 _CXI 10
			109 _LBJ 7
			110 _JBNZ 5
			132 _MLB 5
38		FS	PUT_SRES
9		FS	REPEAT_SEQUENCE 3 CP MAIN LOOP 22
26		FS	REPLACE_CHECK 28 UPDATE_REPLACE 7

NSHC AN/UYM-7 (GP)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
44		FS	ROUND_UP 68 FD 34 39
119		FS	SCT
29		FS	SET_CD1 101 C 4
			103 CM 6
			104 CG 6
			133 HC 4
			135 HCM 6
			136 HCB 6
30		FS	SET_CD2 102 CL 5
			134 HCL 5
20		FS	SPR_CHECK 4 I_SEQUENCE 11
			13 JUMP_ADDRESS 15
			14 OP_READ 19
			15 OP_STORE 21
			16 IA_SEQUENCE 9
			55 AR 9
22		FS	UPDATE_REPLACE 45 CB 4
			49 SC 5
			50 MS 6

NSMC AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
			51 _XOR 4
			52 _ALP 6
			53 _LLP 14
80		FS	_AA
85		FS	_AB
71		FS	_AEI
52		FS	_ALP
79		FS	_ANA
86		FS	_ANB
99		FS	_BC
92		FS	_BZ
101		FS	_C
102		FS	_CL
103		FS	_CH
54		FS	_CNT
100		FS	_CXI
98		FS	_D
63		FS	_DC
107		FS	_DJNZ
106		FS	_DJZ
61		FS	_DL
59		FS	_DS
65		FS	_FA

AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

NSWC

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
66		FS	_FAM
68		FS	_FD
67		FS	_FM
108		FS	_F51
133		FS	_HC
136		FS	_HCB
134		FS	_HCL
135		FS	_HCM
125		FS	_HCP
			126 _HDCP
129		FS	_HD
126		FS	_HDCP
122		FS	_HDL
124		FS	_HDSF
132		FS	_HLB
121		FS	_HLC
128		FS	_HM
127		FS	_HMR
139		FS	_MPI
130		FS	_HRT
123		FS	_HSF
137		FS	_HSIM
138		FS	_MSTC

MSWC

AN/UYK-7 (CP)
INDEX TO FLOW SEGMENTS

14 DEC 79 PAGE 150.017

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
140		FS	_H776
73		FS	_IO
70		FS	_IPI
74		FS	_IR
110		FS	_JBNZ
105		FS	_JEP
112		FS	_JL
111		FS	_JS
113		FS	_J530
114		FS	_J531
115		FS	_J532
76		FS	_LA
84		FS	_LB
109		FS	_LBJ
64		FS	_LBMP
78		FS	_LCIF
72		FS	_LIM
53		FS	_LLP
83		FS	_LM
82		FS	_LNA
81		FS	_LSUM
77		FS	_LXB
95		FS	_P

NSWC AN/UYK-7 (CPI)
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
50		FS	_MS
48		FS	_OR
53		FS	_RA
27		FS	_REPLACE 26 REPLACE_CHECK 5
94		FS	_RI
75		FS	_RP
88		FS	_SA 89 _SXB 3
87		FS	_SB
49		FS	_SC
58		FS	_SDIF
56		FS	_SLP
91		FS	_SM
90		FS	_SNA
57		FS	_SSUM
89		FS	_SXB
60		FS	_TSF
51		FS	_XDR
55		FS	_XP
69		FS	_XS

DISTRIBUTION

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 23209
ATTN: William Carlson

U.S. Naval Electronics Systems Command
Washington, DC 20360
ATTN: John Machado (Code 330)

Director
U.S. Army TRADOC System Analysis Activity
White Sands Missile Range, NM 88002
ATTN: ATAA-SL (Technical Library)

Commander
Naval Ocean Systems Center
271 Catalina Boulevard
San Diego, CA 92152
ATTN: Russ Evers (Code 5200)

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314

(12)

Defense Printing Service
Washington Navy Yard
Washington, DC 20374

Library of Congress
Washington, DC 20540
ATTN: Gift and Exchange Division

(4)

RADC/ISCA
Griffiss Air Force Base
Rome, NY 13441
ATTN: Armand Vito

DMA/STP
Bldg. 56, US Naval OBS
Washington, DC 20305
ATTN: Annette Krygiel

PCDSSA Dam Neck
Virginia Beach, VA 23461
ATTN: Cary D. Upshur (Code 6321)

(2)

PCDSSA Code 6
San Diego, CA 92147

NAVSEC
Code 6122
Department of the Navy
Washington, DC 20362
ATTN: Joe Mallonee

NUSC
Code 313
New London, CONN 06320
ATTN: Dr. Charles Arnold

NCR E&M--SD
16550 W. Bernardo Dr.
San Diego, CA 92127
ATTN: Leslie Stevens
Manager, Product Firmware

NANODATA
6065 Madra Ave.
San Diego, CA 92120
ATTN: Robert C. Boe

NANODATA
One Computer Park
Buffalo, NY 14203

Sperry Univac
Univac Park
PO Box 3525
St. Paul, MN 55165
ATTN: Doug Wiedenman
M.S. U2S17

SDC
601 Caroline St.
Fredericksburg, VA 22401

(2)

Hale Associates Research Corp.
PO Box D.J.
Stony Brook, NY 11790
ATTN: John Hale

USC/Information Sciences Institute
4676 Admiralty Way
Marina Del Ray, CA 90291

Defense & Space Systems Group of TRW, Inc.
One Space Park
Redondo Beach, CA 90278
ATTN: Barry Pross

Naval Sea Systems Command
Washington, DC 20362
ATTN: Lowell Wood

GIDEP Operations Office
Corona, CA 91720

Local:

E41	
K60	
K61 (Carroll Shelton, Henry Walker)	(2)
K70	
K71	
K74	(40)
N30 (R. Hein)	
N52 (Hubbard)	
X210	(6)
X211	(2)