NAWCWD TP 8823
Revision 1

Coupling Considerations in Assembly Language

by
Anthony S. Cantone
Systems Safety Engineering Branch
Systems Engineering Department

FEBRUARY 2018

NAVAL AIR WARFARE CENTER WEAPONS DIVISION
CHINA LAKE, CA 93555-6100

Naval Air Warfare Center Weapons Division

FOREWORD

This report discusses coupling issues arising in assembly language source code, as contrasted to similar issues arising in high order language (HOL) source code. Although there are many sources for coupling, this report focuses on data and control coupling because many projects at the Naval Air Warfare Center Weapons Division (NAWCWD) require compliance with DO-178B and DO-178C, which contain guidelines relating to data and control coupling. The projects under consideration are ones where new code is added to a large body of existing legacy code. A coupling measure employed in the literature is suggested for measuring the coupling of code. The coupling measure of the existing legacy code is considered unknown, but its effect on development and maintenance will serve as indicators on whether the coupling measure is high or low. Both cases are considered, and a strategy to address the coupling concerns is formulated.

This report was reviewed for technical accuracy by Jia Y. Chen (Code 512Y00D) and Michael Brush (Code 494700D).

Approved by
H. D. KOOIMA, *Head*
Systems Engineering Department
5 January 2018

Under authority of
B. K. COREY
RDML, U.S. Navy
Commander

Released for publication by
J. L. JOHNSON
Director for Research and Engineering

NAWCWD Technical Publication 8823, Revision 1

Published by ..... Technical Communication Office
Collation ..... Cover, 7 leaves
First printing ..... 7 paper, 8 electronic media

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
13-02-2018	Final	1 November 2017 – 19 December 2017

4. TITLE AND SUBTITLE
Coupling Considerations in Assembly Language (U)

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)
Anthony S. Cantone

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Air Warfare Center Weapons Division
1 Administration Circle
China Lake, California 93555-6100

8. PERFORMING ORGANIZATION REPORT NUMBER
NAWCWD TP 8823, Revision 1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
None

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
N/A

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
None

14. ABSTRACT

(U) This report discusses coupling issues arising in assembly language source code, as contrasted to similar issues arising in high order language (HOL) source code. Although there are many sources for coupling, this report focuses on data and control coupling because many projects at the Naval Air Warfare Center Weapons Division (NAWCWD) require compliance with DO-178B and DO-178C, which contain guidelines relating to data and control coupling. The projects under consideration are ones where new code is added to a large body of existing legacy code. A coupling measure employed in the literature is suggested for measuring the change in coupling due to code added to the existing legacy code. The coupling measure of the existing legacy code is considered unknown, but its effect on development and maintenance will serve as indicators on whether the coupling measure is high or low. Both cases are considered, and a strategy to address the coupling concerns is formulated.

15. SUBJECT TERMS
Assembly Language, Control Coupling, Coupling Measure, Data Coupling, DO-178B, DO-178C, High Order Language (HOL)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Anthony Cantone
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	14	19b. TELEPHONE NUMBER *(include area code)* (760) 939-1820

REVISION HISTORY

Rev	Date	Change Description
-	January 2018	Initial release
1	February 2018	Foreword: “change in coupling due to code added to the existing legacy code.” becomes “coupling of code.” Page 3, Section 2.0: “entities.” becomes “code entities.” Page 3, Section 3.0: “one must know about one object or another,” becomes “a module must know about another,” Page 6 after the 2nd paragraph of Section 6.0: Added new paragraph: “An assumption made in this report is that there is an unwillingness to modify legacy code to mitigate any existing coupling problems. This report will concentrate instead on the best strategy to mitigate any increase in coupling due to the added code.” Page 9, Section 7.2 in the paragraph following Equation (7): replace “protecting” with “mitigating.” Page 10, Section 7.2 in the paragraph following Equation (12): replace “protecting” with “mitigating.” Page 10, Section 8.0 in the last paragraph denoted 2.: “by difficulty” becomes “by a lack of difficulty.”

This page intentionally left blank.

CONTENTS

This page intentionally left blank.

1.0 SCOPE

This report discusses coupling issues arising in assembly language source code, as contrasted to similar issues arising in high order language (HOL) source code. Although there are many sources for coupling, this report focuses on data and control coupling because many projects at the Naval Air Warfare Center Weapons Division (NAWCWD) require compliance with DO-178B and DO-178C (References 1 and 2), which contain guidelines relating to data and control coupling.

2.0 WHAT IS COUPLING?

Coupling is generally defined as a relationship between code modules such that

1. A change in one module forces a change in another module. For example, if a certain module uses a variable defined in another module, and the type of that variable is changed in one of the modules, the method of handling that variable in the other module may be affected. This becomes obvious during maintenance.

2. Testing a function in one module necessitates inclusion of functions in another module. For example, testing a function that solves a partial differential equation using finite elements requires the inclusion of a matrix inversion routine.

More briefly, coupling is a measure of the number of dependent interfaces among code entities.

Coupling can be loose or low, or it can be tight or high. What is meant by low or high coupling is discussed in Section 5.0, “Coupling Measure.”

3.0 WHY IS TIGHT OR HIGH COUPLING UNDESIRABLE?

Coupling issues arose when structured design was first investigated. Tight coupling was an early indication of poor quality (Reference 3). Some coupling is unavoidable. Modules pass data and control flags to each other, and databases are sometimes used as sources of data for many modules. However, excessive coupling results in dependencies that are inconvenient during development or maintenance, when changes in one module ripple into changes in other modules.

In general, the more facts a module must know about another, the tighter the coupling. Problems arise when these facts change.

4.0 WHAT TYPES OF COUPLING CAN BE IDENTIFIED?

4.1 CONTENT COUPLING

One of the most severe forms of coupling takes place when a module modifies local data or code in another module. The term "pornographic programming" is sometimes used to refer to the practice of dynamically changing code. This can easily be done in assembly languages—just write a command that modifies the opcode of another command.

A module modifying (or referring to) local data in another module is another form of content coupling. Because of the difficulty of information hiding in assembly language, it is trivial to cross the boundary into the realm of data local to a particular area of assembly language code.

Content coupling also occurs when a module branches to a local label of another module. In assembly language, the concept of "local" is not well-defined. Nevertheless, the intent for the use of the label should be clear, and from that, the concept of a local label can be well-defined.

In a HOL, a branch from one function to a label inside another function, if enabled, would be an example of nonstructured programming and is not generally allowed. Content coupling issues can arise.

4.2 COMMON COUPLING

Global data sharing among modules is a form of common coupling. This is particularly a challenge with assembly languages, where every data element can be considered as a global variable. However, consider the approach described in Section 7.1, "Assembly Language Modules."

4.3 CONTROL COUPLING

Next down the coupling severity ladder is control coupling, where a module controls the sequence of processing in another module by passing control flags as parameters or via global variables. In assembly languages, this is usually done by setting bits in registers that are used to control processing.

4.4 STAMP COUPLING

Stamp coupling is similar to what is encountered in common coupling with the sharing of global data. However, selectively sharing the global data mitigates the negative

effects of global data sharing. Selective sharing of global data requires information hiding and is possible to the limited extent in assembly language discussed in Section 7.1, "Assembly Language Modules."

4.5 DATA COUPLING

The passing of data items via the use of parameter lists probably contributes most often to coupling issues. Nevertheless, this is much preferred to sharing global data. In assembly languages, every variable is global. Employing a pseudo module approach, where variables are used only locally (say within a section resembling a class) helps mitigate this problem. See Section 7.1, "Assembly Language Modules."

4.6 EXTERNAL COUPLING

Nearly the weakest form of coupling is the sharing by modules of external formats, protocols, and interfaces, known as external coupling.

4.7 MESSAGE COUPLING

The weakest form of coupling arises when messages are passed between modules. Examples of message coupling: Dependency Injection and Observables (Reference 4).

5.0 COUPLING MEASURE

A useful coupling measure is found in Reference 5. This is based on a technical note written by M. A. Hennell with the title "Data Coupling and Control Coupling." Dr. Hennell's measure is as follows:

$$C = 1 - \frac{1}{di+2ci+do+2co+gd+2gc+w+r} \quad (1)$$

where

- C = degree of software coupling (low coupling ~0.66, high coupling ~1.0)
- di = number of input data parameters
- ci = number of input control parameters
- do = number of output data parameters
- co = number of output control parameters
- gd = number of global variables used as data
- gc = number of global variables used as control
- w = number of modules called (fan-out)
- r = number of modules calling the module under consideration (fan-in)

These definitions are easy to understand in the environment of a HOL but need interpretation when using assembly language. For example, if the approach discussed in Section 7.1, “Assembly Language Modules” is used to affect information hiding, then a “control parameter” is any variable that is defined outside the pseudo-class (see Section 7.1, “Assembly Language Modules”) and is used to make a decision in the code appearing in the class.

6.0 HIGH COUPLING MITIGATED

Reference 4 points out what can be done with some of the coupling types discussed in Section 4.0, “What Types of Coupling Can Be Identified?” Summarizing Reference 4, we have the following mitigations:

- Content Coupling: encapsulation
- Common Coupling: introduce abstractions
- External Coupling: eliminate knowledge of formats from the domain
- Control Coupling: use strategies or states
- Stamp Coupling: pass actual data
- Data Coupling: employ message passing

The Law of Demeter is also relevant in mitigating coupling. A *Wikipedia* page summarizes the fundamental notion (Reference 6):

> A given object should assume as little as possible about the structure or properties of anything else (including its subcomponents), in accordance with the principle of information hiding.

An assumption made in this report is that there is an unwillingness to modify legacy code to mitigate any existing coupling problems. This report will concentrate instead on the best strategy to mitigate any increase in coupling due to the added code.

7.0 HOW DOES THIS APPLY TO ASSEMBLY LANGUAGE PROGRAMMING?

7.1 ASSEMBLY LANGUAGE MODULES

Modules that arise when coding in a HOL make possible information hiding, which mitigates coupling issues. It is possible to simulate the concept of module in assembly language code despite the flat nature of the language, as explained in the following paragraphs. Information hiding can be simulated in an assembly language environment as follows:

1. Divide the source code into sections, each of which architecturally resembles a class in a HOL such as C++. These are the assembly language "modules." The source code in these modules is analogous to methods found in classes, hence the term "pseudo-class." Every variable in assembly language code is by default a global variable, but if the use of every variable found in a particular section is restricted to that section, then effectively there are no global variables. Do the same for control variables (variables used in decisions). Where it is necessary for a module to refer to a variable outside the class, whether for data or control, this can be regarded as parameter passing.

2. Publish for coder reference only those names that are part of the module's interface to outside source code.

3. In peer review checklists, call attention to the possibility of data or control variables being used indiscriminately as global variables.

7.2 EMPLOYING THE COUPLING MEASURE

In particular, how can we apply Dr. Hennell's measure when modifying legacy code, assuming both the legacy code and the modification are written in an assembly language? One possible approach is as follows:

a. An agreement is made not to make any attempt at remedying any high coupling situation that may exist in the legacy code.

b. Given any variable in Dr. Hennell's coupling measure in Section 5.0, "Coupling Measure," rewrite the coupling measure, lumping the variables not involving the variable under consideration into a term "*D*." As an example, suppose the effect of changing the variable gd (number of global variables used as data) on the coupling measure is desired. Then the coupling measure equation becomes

$$C = 1 - \frac{1}{D+gd} \tag{2}$$

c. The temptation to compute the partial derivative of *C* with respect to the element under consideration to calculate the effect that a change in this variable will have on the software coupling should be resisted:

$$\frac{\partial C}{\partial gd} = \frac{1}{(D+gd)^2} \tag{3}$$

Differentials are approximations that are valid only for small changes in the independent variable, in this case gd. "Small" means the fraction $\Delta gd/gd$ should be small with respect to 1, which is not the case because gd will be assumed 0 later. Therefore, the accurate difference equation will be used:

$$C + \Delta C = 1 - \frac{1}{D+gd+\Delta gd} \tag{4}$$

This difference equation is obtained by incrementing the variable under consideration, gd by the amount Δgd. The resulting change in *C* is ΔC. Therefore, the pair (gd + Δgd, C + ΔC) satisfies Equation 2. Then Equation 4, the change ΔC in the coupling measure *C*, is easily obtained when the above pair is substituted in Equation 2, and the equation is solved for ΔC.

There are many ways to express this difference equation. Equation 4 is as good as any. This last equation should tell us the effect on the coupling due to a change in the number of global variables Δgd used as data. However, it assumes two things are known: the value *C* of the coupling of the existing (i.e., legacy) code, and the number of global variables *gd* in the unmodified code. These are needed to get the value of *D*.

Generally, one has a sense of whether there are coupling problems with the preexisting code. However, even if there is little information about coupling issues, there is a way to estimate the effect of adding more global variables. Two examples will show this. The first example deals with a legacy assembly code that is easy to modify and test, leading to the conclusion that there are no coupling issues (*C* = 0.70, for example). Let us assume that there are no variables that we can call global (*gd* = 0) because of the approach discussed in Section 7.1, "Assembly Language Modules." We wish to find the effect on the coupling measure *C* that adding two global variables would produce. We can estimate the value of *D* as follows from Equation 2, using C = 0.70 and *gd* = 0:

$$D = \frac{1}{1-C} - gd = 3.33 \tag{5}$$

Using $\Delta gd = 2$, the change in the coupling measure is calculated from Equation 4 as

$$C + \Delta C = 1 - \frac{1}{3.33+0+2} = 0.8084 \tag{6}$$

which is a significant increase in coupling, from 0.70 to 0.81. Thus, we can make the observation here that if the legacy coupling measure is low, it is easy to increase it to an undesirably high value.

The second example considers a legacy where development and maintenance problems make it obvious that there are coupling issues. A change in the type of a variable has a rippling effect throughout the code. Attempts to test a small part of the code are not successful unless a large amount of code is added to the test image. So we assume a high coupling measure: C = 0.90, and no global data (the high C has another cause): $gd = 0$, then D = 10. Adding two global variables: $\Delta gd = 2$, the change in the coupling measure is

$$C + \Delta C = 1 - \frac{1}{10+0+2} = 0.9167 \tag{7}$$

We see it is difficult to make an already bad situation worse. The best strategy seems to be to concentrate on not making a good situation bad, at the same time not bothering about mitigating an already bad situation.

The process for global variables used as control variables is similar, beginning with the following equation:

$$C = 1 - \frac{1}{D+2gc} \tag{8}$$

or, as a difference equation:

$$C + \Delta C = 1 - \frac{1}{D+2gc+ 2\Delta gc} \tag{9}$$

where D is defined appropriately from Equation 1. We wish to find the effect on the coupling measure C that adding two control variables would produce. Estimating the value of D from Equation 8, using C = 0.70 and gc = 0:

$$D = \frac{1}{1-C} - 2gc = 3.33 \tag{10}$$

Using Δgc = 2, the coupling measure becomes, from Equation 9,

$$C + \Delta C = 1 - \frac{1}{3.33+0+2(2)} = 0.8636 \tag{11}$$

which again is a significant increase in coupling from 0.70 to 0.86. On the other hand, if the coupling is already high, say C = 0.9, and no control variables (gc = 0), then D = 10 from Equation 10. Adding two control variables Δgc=2, the coupling measure becomes

$$C + \Delta C = 1 - \frac{1}{10+0+2(2)} = 0.9286 \quad (12)$$

and we arrive at the same plan of action for a change in the coupling measure for control variables as we had with global variables: The best strategy seems to be to concentrate on not making a good situation bad, while at the same time to not bother mitigating an already bad situation.

8.0 SUMMARY

The effect on coupling by adding global and control variables to legacy code programmed in assembly language is determined. The existing degree of coupling in the legacy code determines the plan of action for the new additions. Considerations are

1. A coupling measure originated by M. A. Hennell with the title “Data Coupling and Control Coupling” is found to be useful in determining the effect on the coupling measure due to added global or control variables.

2. The existing degree of coupling of the legacy code determines whether it is worthwhile to take mitigating actions to guard against an increase in the coupling measure in added code. If the coupling measure in the legacy code is thought to be low, as determined by a lack of difficulty in development and maintenance, just a few global or control variables in the added code can increase this measure significantly, so mitigations are advised. If the coupling measure of the legacy code is thought to be high, added global or control variables will not affect it significantly, so mitigation measures applied to the added code will not result in a significant change in the worsening of the coupling measure.

REFERENCES

1. RTCA, Inc. “Software Considerations in Airborne Systems and Equipment Certification.” Washington, D.C., 1 December 1992. (DO-178B.)

2. RTCA, Inc. “Software Considerations in Airborne Systems and Equipment Certification.” Washington, D.C., 5 January 2012. (DO-178C.)

3. Martin Fowler. Last accessed 6 November 2017, http://martinfowler.com. Email is fowler@acm.org.

4. “Thoughts on Coupling in Software Design,” by Ioan Fagarasan, posted 25 July 2016, accessed 6 November 2017. https://codurance.com/software-creation/2016/07/25/thoughts-on-coupling-in-software-design/.

5. Wikipedia. “Coupling,” last edited 26 October 2017, accessed 6 November 2017, https://en.wikipedia.org/wiki/Coupling_(computer_programming).

6. Wikipedia. “Law of Demeter,” last modified 24 October 2017, accessed 6 November 2017, https://en.wikipedia.org/wiki/Law_of_Demeter.

This page intentionally left blank.

INITIAL DISTRIBUTION

1 Defense Technical Information Center, Fort Belvoir, VA
2 DCS Corporation, Ridgecrest, CA (Henderson, K.)

ON-SITE DISTRIBUTION

2 Code 4F0000D (archive copies)
2 Code 4G0000D (file copies)
6 Code 416100D
  Cantone, A. (2)
  Chirkis, K. (2)
  Merrill, R. (2)
2 Code 494700D, McCue, G.