MIPS Documentation and Style Guide

CS/SE 3340 Computer Architecture

Dr. Karen Mazidi

What makes good code good:

- ✓ Code logic should be easy to read
- ✓ Code should be modular
- ✓ Code style should be internally consistent
- ✓ Code should be documented internally and externally
- ✓ Code should be tested and verifiable
- ✓ Code should be efficient
- ✓ When you write code, think of the person who will have to maintain it.

Code Style.

What we call *style* generally involves things like use of white space, comments, naming identifiers, etc. A "Hello World" program is on the next page. Notice the following features.

- The file name HelloWorld.asm indicates the purpose of the program and ends in .asm
- A header comment block gives program information such as author, date, purpose
- Comments at the bottom of the main function show a sample run, or test case
- In-line comments are lined up vertically (not always possible)
- There is whitespace between the two main sections of code: .data and .text
- Label names for variables and code points are in their own vertical space on the left, all lower case; identifier names should be consistent in either camelCase or underscore_case
- Code is double indented: opcode and operand
- Each action of the code has its own comment to the side and is separated from other actions by a blank line

If I want to figure out what this code does a year from now, I can just scan the header comment block and the inline comments down the right and finally look at the sample run comments and I've got it. I can always drill down to the details as needed.

```
1  ##############################################################
2  # HelloWorld.asm
3  # author: Karen Mazidi
4  # created: 2018.5.3
5  # purpose: demo console I/O
6  ##############################################################
7
8          .data
9  name:   .space          20
10 msg1:   .asciiz         "What is your name? "
11 msg2:   .asciiz         "Hello "
12
13         .text
14 main:   li      $v0, 4          # print name prompt
15         la      $a0, msg1
16         syscall
17
18         li      $v0, 8          # read name
19         la      $a0, name
20         li      $a1, 20
21         syscall
22
23         li      $v0, 4          # print hello
24         la      $a0, msg2
25         syscall
26
27         li      $v0, 4          # print name
28         la      $a0, name
29         syscall
30
31         li      $v0, 10         # exit
32         syscall
33
34 # sample run
35 # What is your name? Karen
36 # Hello Karen
```

External Documentation.

Internal documentation as discussed above is useful for other programmers to read and understand your code. For a large project, however, we need external documentation a swell which includes things such as flow charts, UML diagrams, requirements documents, a user manual, legal info and more, depending on the environment. There are specialized external documentation programs like Sphinx (Python, C, C++), Doxygen (C++, PHP, Java, Python), and many more. You should learn at least one of those for your projects in other languages.

For our purposes in assembly language, we can keep things pretty simple. A Word document can be used to create a doc or pdf. Another option is to use markdown. Next, we show a simple example of external documentation in Markdown for the HelloWorld program, but first: What is Markdown? Markdown is a simple markup language that lets you format as you type without taking your keys off they keyboard. It is usually a plain text file ending in .md that can be rendered into html.

Markdown Using Atom.

There are a lot of free editors for Markdown but I'm going to discuss Atom since it is a common code editor, it is cross platform, and free. You can download atom here: https://atom.io/

Installation should be painless. Open Atom and start typing, save your document as .md

When you want to preview what the markdown will look like: ctrl-shift-m

To export the html, right-click on the markdown preview and choose save as html

If you want to change the default style to the GitHub style, Go to Packages -> Markdown -> toggle GitHub style

Markdown Basics.

Markdown cheat sheet: https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf

What kinds of things can you do in markdown?

- Format headings
- Format text italic or bold
- Make lists (ordered or unordered)
- Place images
- Create links
- Show code blocks
- Tables

Markdown Example.

Below, we have a screen shot from within Atom showing our markdown code on the left and the rendered markdown on the right. Notice the following:

- Headers of varying sizes starting with #
- Bold text is surrounded by ** two stars**
- Italic text is surrounded by *one star*
- We can make ordered lists with * and indents
- We can insert a code block by starting and ending with 3 consecutive backticks, the first 3 backticks are followed by the code type, in our case we say assembly