Assembler Style Guide

1 Overview

From henceforth, your code will comply with the following style guide. Why? Because code readability and program understanding is important. A significant component of readability and understanding comes from consistent formatting, appropriate selection and use of labels (i.e., variable and code target names), and appropriate use of comments. Get in the habit now of writing good, self-documenting code; your future coworkers and supervisors will thank you for it!

2 Author Block

Every source code file that you submit will start with an author block. The author block will contain your name, the file name, the date, and a brief description of the project/file. The author block is left flush. The following is an example:

```
; Author: Mike Jochen
; File: somethingCool.asm
; Date: 1/1/1970
; Description: This project computes the elusive, fictional value
; of the left-handed Euler totient. It is really cool and well written.
```

3 Segment Names

Segment names are left flush. Only use segments that actually contain something. Foor example, if segment `.bss` is empty, don't use it. The following is an example:

```
segment .data
; Write your initialized data here . . .
; blah blah . . .

segment .text
     ; Cool instructions go here
```

4 Label Names

All labels in program code (i.e., jump targets and subprogram names) are left flush. Labels for variable names are indented one tab (five spaces). In CPSC 232, use camelCase notation for label names:

- The first character is a lowercase letter
- SubSequent characters may be numeric of lower-case alphabetic
- If the label consists of multiple words, smash the words together (no space between them) and capitalize the first letter of each subsequent word.
- Constants are all capitals. If a constant value consists of multiple words, use the underscore character to separate words.

5 Program Text/Instructions

Program instructions are indented one tab (five spaces). Separate operands from opcodes by at least two spaces. Operands will align vertically in the code. Separate multiple operands to the same instruction with one space.

6 Comments

Use comments to explain/document your code. In-line comments will immediately follow an instruction. In-line comments will align vertically in the code. Block comments are left flush. Use block comments to explain subprograms (methods or functions) or blocks of code that require greater documentation than an in-line comment would provide.

7 Example

```
; Author: Mike Jochen
; File: hello.asm
; Date: 1/1/1970
; Description: Print out Hello World! - our very first assembler program.

%include asm_io.inc

     NINE equ 9    ; a numeric constant

segment .data
     sayHello    db    "Hello Assembly World!",10,0

segment .text
     global  asm_main

; Here is our main
; the start of the program
asm_main:
     enter   0,0
     pusha

     mov     eax, sayHello
     call    print_string
     popa
     mov     eax, 0
     leave
     ret
```