

SPIM16 EECS Project 1 Specification

ASSEMBLY APPLICATION FORMAT

Assembly File (.asm)

- Program Lines
 - Zero or One Label, plus
 - Zero or One Instruction, plus
 - Zero or One Comment
- Labels – Symbol Definitions
 - Starts with a letter, plus
 - Zero or more symbol characters, plus
 - A single colon ':'
- Symbol Characters
 - Letter
 - Digit
 - Underscore
- Instructions
 - Starts with a name, plus
 - One or more operands separated by whitespace
- Operands (depends on instruction format)
 - A Register, OR
 - An Immediate Value, OR
 - A Label
- Comments
 - Begin with pound sign '#'
 - Continues to end of line
- Load/Store
 - Source/destination register first operand
 - Immediate value second operand
 - Base address register third operand surrounded by parenthesis

All assembler input is to be considered valid and well formed.

SAMPLE APPLICATION

```
# Sample Application
Main:  # Main Function
      and $s1, $s1, $zero # clear contents
      addi $s1, $zero, Count # handle appropriately
      lw  $s0, 0($s1) # data at Counter address
Loop:  bz  $s0, Done # test
      or  $a0, $s1, $zero # move
      addi $a0, $a0, 2 # next instruction
      addi $ra, $zero, Loop
      addi $ra, $ra, 12 # return address
      jr  $a0 # jump to Dummy
      addi $s0, $s0, -1 # decrement count
      sw  $s0, 0($s1) # store value
      j   Loop
Done:
      # end here
      j   Done
      addi $s0, $zero, 43
      addi $s1, $zero, 34
Count: .data 17 # data value
Dummy: # does nothing
      jr  $ra # return
      j   Foobar # undefined
      .data -7
```

Usage

assembler filename

BINARY INSTRUCTION FORMATS

Register Format (R)

```
# Assembly Code
operation rd, rs, rt

# Machine Code
oooo ssss tttt dddd
```

Register-Immediate Format (RI)

```
# Assembly Code
operation rt, rs, immediate
operation rt, immediate(rs)

# Machine Code
oooo ssss tttt iiii
```

Immediate Format (I)

```
# Assembly Code
operation rs, immediate
operation rs, label

# Machine Code
oooo ssss iiii iiii
```

Jump Format (J)

```
# Assembly Code
operation immediate
operation label

# Machine Code
oooo iiii iiii iiii
```

LABEL HANDLING

Label Addressing

- Label Address is Address of Next Instruction
- Only Instruction Lines Increment Addresses
- Addresses Increment by Instruction Width in Bytes (2)
- Store All Label Definitions in a Symbol Table

Symbol Table

- Table Entry (struct Symbol)
 - Symbol Label (char*)
 - Defined? (char, 'y' | 'n')
 - Address Value (unsigned int)
 - Usage List (struct Use*)
- Usage Entry (struct Use)
 - Instruction (char*)
 - Usage Address (unsigned int)
- Branch instructions are ignored

FILE OUTPUT

- Symbols File (.sym), Object File (.o), RAM File (.ram)
- The .ram file copy the object file in hexadecimal text
- Symbol addresses are hexadecimal and 4 characters wide
- Symbol file lines have one definition followed by the usage list separated by tabs per line
- Code project in ANSI-C. See my ANSI-C tutorial for help