Lecture Review and Textbook References

- Section 2.5 – Translating assembly language into machine language
  - Page 85, Figure 2.6 has excellent examples of translating assembly language instructions to machine language for both R and I format instructions
- What are MIPS fields? What do they contain?
- What is a pseudo-instruction? How can it be represented in machine language?
- How many bits are MIPS instructions?

R Format Instructions:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

I Format Instructions:

op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

op – operation code
rs/rt – first and second source registers
rd – destination register
shamt – shift amount
funct – function code

J Format Instructions:

op	(pseudo) address
6 bits	26 bits

R Format Instruction Example: `add $s1, $s2, $s3`

op	rs	rt	rd	shamt	funct
0	18	19	17	0	32

I Format Instruction Example: `addi $s1, $s2, 100`

op	rs	rt	constant or address
8	17	18	100

op – operation code
rs/rt – first and second source registers
rd – destination register
shamt – shift amount
funct – function code

J Format Instruction Example: `j 10000`

op	(pseudo) address
2	10000

Corrected Error 28-Jan-2016

R Format Instruction Example: `add $s1, $s2, $s3`

op	rs	rt	rd	shamt	funct
000000	10010	10011	10001	00000	100000

I Format Instruction Example: `addi $s1, $s2, 100`

op	rs	rt	constant or address
01000	10001	10010	0000000001100100

op – operation code
rs/rt – first and second source registers
rd – destination register
shamt – shift amount
funct – function code

J Format Instruction Example: `j 10000`

op	(pseudo) address
00010	00000000000010011100010000

More about this next week…

Corrected Error 28-Jan-2016

Terminal and Non-Terminal Functions

- Wikipedia definition
  - a non-terminal function is a function (node) in a parse tree which is either a root or a branch in that tree whereas a terminal function is a function (node) in a parse tree which is a leaf
    - Source: https://en.wikipedia.org/wiki/Terminal_and_non-terminal_functions Retrieved 21-Jan-2016
- Informal definition
  - a terminal function is a function which does not call another function, a non-terminal function is one that does

Lab Activity

Programming in MIPS

Download tutorial3starter.s from the course Moodle. We will add to the same file incrementally.

Step 1

Write a terminal function to read and print a string using a statically allocated buffer.

Step 2

Create a terminal function which returns the length of a string

Hint: This will involve loops and indexing into a character array (like C strings)

Step 3

Create a terminal function to convert a string to uppercase

Hint: This will involve simple comparisons, ASCII character codes and the load/store byte instructions

Step 4

Create a **non-terminal** function to dynamically allocate an array to store a histogram of character frequencies, then compute and print that histogram.

Hint: Your function is non-terminal because it will call 1 or more terminal (helper) function(s). What complication does this introduce?

Corrected Error 28-Jan-2016

Completed Program Sample Output

```
Enter a string: CMPT 215 is awesome!
CMPT 215 is awesome!
the string contains 21 characters
in upper case, the string is CMPT 215 IS AWESOME!
the count for A is 1.
the count for B is 0.
the count for C is 1.
the count for D is 0.
the count for E is 2.
the count for F is 0.
the count for G is 0.
the count for H is 0.
the count for I is 1.
the count for J is 0.
the count for K is 0.
the count for L is 0.
the count for M is 2.
the count for N is 0.
the count for O is 1.
the count for P is 1.
the count for Q is 0.
the count for R is 0.
the count for S is 2.
the count for T is 1.
the count for U is 0.
the count for V is 0.
the count for W is 1.
the count for X is 0.
the count for Y is 0.
the count for Z is 0.
```