CS61c Summer 2014 Discussion 5 – Everything is a Number!

July 7, 2014

1 MIPS Instruction Formats

Every MIPS instruction is represented with 32 bits! They come in three formats:

- R-Instruction format (register-to-register) Examples: `add, and, sll, slt, jr`

opcode	**rs**	**rt**	**rd**	**shamt**	**funct**
6 bits	5 bits	5 bits	5 bits	5 bits	6bits

- I-Instruction Format (register immediate) Examples: `addiu, andi, beq, bne`

opcode	**rs**	**rt**	**immediate**
6 bits	5 bits	5 bits	16 bits

- J-Instruction Format (jump format) For `j and jal`

opcode	**address**
6 bits	26 bits

Here's what each field in the formats means:

opcode	Indicates operation, or arithmetic family of operations (for opcode 0, which is R-type)
funct	Indicates specific operation within arithmetic family of operations
rs, rt, rd	For R-type, rs and rt are sources with rd as destination - rules vary for other formats!
shamt	Shift amount for instructions that perform shifts
immediate	Relative address or constant, will be 0 or sign-extended to 32 bits
address	Absolute address

See the MIPS Green Sheet for more details!

Exercise 1. How many total possible instructions can we represent with this format?

Exercise 2. What could we do to increase the number of possible instructions?

2 Decoding and Encoding MIPS Instructions

Exercise 3. Convert **`addi $t1, $t0, 5`** to its HEX representation.

Exercise 4. Decode the following program and describe its function.

Addres	Instruction	Decoded Instruction
0x00	0x0085402A	
0x04	0x11000002	
0x08	0x00A01020	
0x0c	0x03E00008	
0x10	0x00801020	
0x14	0x03E00008	

Exercise 5. Given the following MIPS code (and instruction addresses), fill in the blank fields for the following instructions (you'll need your green sheet!):

```
0x002cff00:  loop: addu $t0, $t0, $t0     | 0 |   |   |   |   |     |
0x002cff04:        jal foo                | 3 |                     |
0x002cff08:        bne $t0,$zero,loop     | 5 | 8 |   |             |
...
0x00300004:  foo:  jr $ra                 $ra= ______________
```