S'more Instructions*

A continuation of discussion of the MIPS ISA, instructions and instruction formats

* Cover Graham cracker with a marshmallow and chocolate

Fact Sheet on Registers/Instructions

- Machines typically 16 or 32 registers
- More registers => larger instruction format
- More registers => possibly slower clock speed
- Register 0 is the constant 0 in MIPS ISA
- Extending the instruction set ...

```
sub $8, $0, $8  # negate $8
add $9, $0, $5  # move $5 to $9
add $5, $0, $0  # set $8 to zero
```

Every computer (ISA) has a Principles of Opertions manual (Prince of Ops); see appendix A.10 for MIPS

R-types Are For Basic Operations

- Different operations are specified by different codings of the **op** field and **funct** field

`(op:funct) rd, rs, rt`

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

	op	funct	
`add`	0	32	addition
`sub`	0	34	subtraction
`and`	0	36	logical and
`mult`	0	24	multiplication
`slt`	0	42	set less than

Example: `and $7, $3, R4  # Bitwise and`

0	3	4	7	0	36
000000	00011	00100	00111	00000	100100

R-types Continued

- Shifting instructions also have 3 operands
- **shamt** is used, but **rs** is unused

`(op:funct) rd, rt, shamt`

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

```
sll      0                0 # shift left
srl      0                2 # shift right
sra      0                3 # arith right
```

Example: `srl $7, $3, 12  # Logical right shift`

0	0	3	7	12	2
000000	00000	00011	00111	01100	000010

Decoding

- Decoding the instruction (second step in the fetch execute cycle):

```
add  $8,  $9   $10
```

000000	01001	01010	01000	00000	100000
0	9	10	8	0	32
op	rs	rt	rd	shamt	funct

What Instruction is all 0's?

```
???  ??,  ??   ??
```

000000	00000	00000	00000	00000	000000

I-type, Immediate Instructions

- When an operand is a small constant it can be stored in the instruction, saving fetching it from memory and using a general register.

```
addi  $8,  $9  100   # Add immediate
```

op	rs	rt	immediate	I-type instructions
8	9	8	100	
001000	01001	01000	0000000001100100	
6	5	5	16	

op	rs	rt	rd	shamt	funct	R-type instructions

Immediate Instructions

Immediate variations exist for most instructions

```
op    rt, rs, immediate
```

`addi`	add immediate
`andi`	and immediate
`ori`	or immediate
`slti`	set less than immediate

Why is there no subi instruction?

op	rs	rt	immediate

I-type instructions

Example: `andi $7, $3, 12  # Logical and`

0000000000000000	0000000000001100

and

register 3 operand

lui, lui

The 16bit limit on immediate data is not severe

load upper immediate (lui) moves its immediate data to the left-most 16-bits; it zeroes the right-most 16-bits; **rs** is unused

```
lui  $8, 100  # Load upper immediate
```

op	rs	rt	immediate
15	0	8	100
001111	00000	01000	0000000001100100

I-type instructions

0000000001100100	0000000000000000

Register 8