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 have 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 … *pseudo instructions*

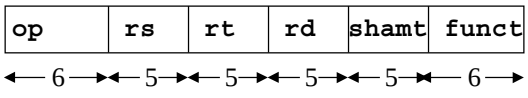
```
sub $8, $0, $8  # negate $8
add $9, $0, $5  # move $5 to $9
add $5, $0, $0  # set $8 to zero
```

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

R-types Are For Basic Operations

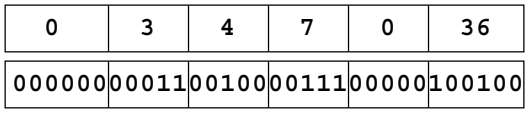
- Different operations are specified by different codings of the **op** field and **funct** field

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



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

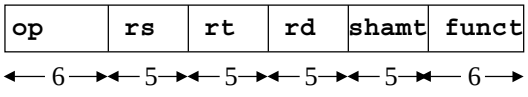
Example: `and $7, $3, R4  # Bitwise and`



R-types Continued

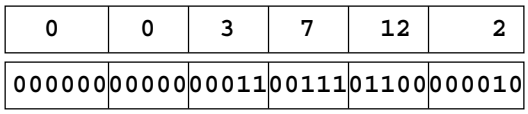
- Shifting instructions also have 3 operands
- **shamt** is used, but **rs** is unused

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



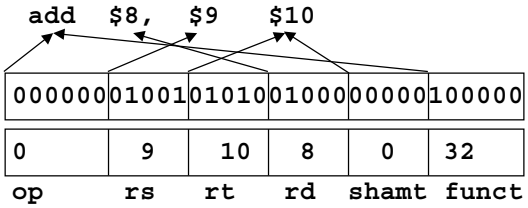
`sll`	`0`	`0 # shift left`
`srl`	`0`	`2 # shift right`
`sra`	`0`	`3 # arith right`

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

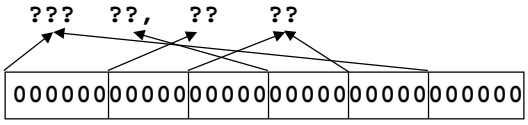


Decoding

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

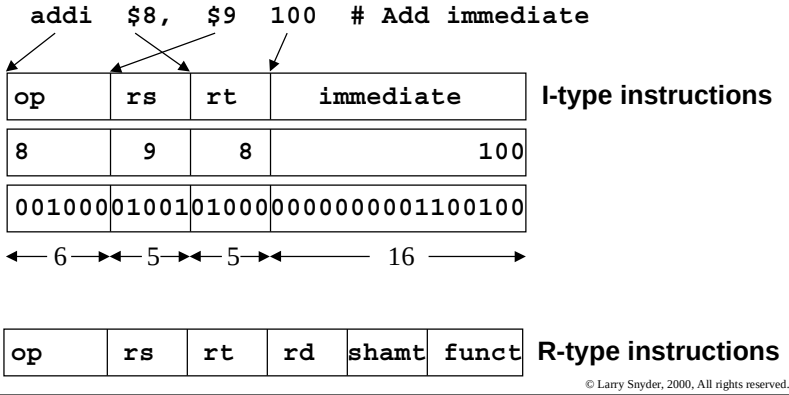


What Instruction is all 0's?



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.



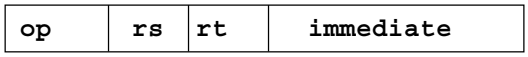
Immediate Instructions

Immediate variations exist for most instructions

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

- **addi** — **add immediate**
- **andi** — **and immediate**
- **ori** — **or immediate**
- **slti** — **set less than immediate**

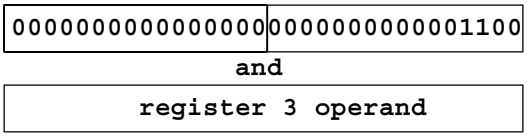
Why is there no **subi** instruction?



I-type instructions

Example:

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

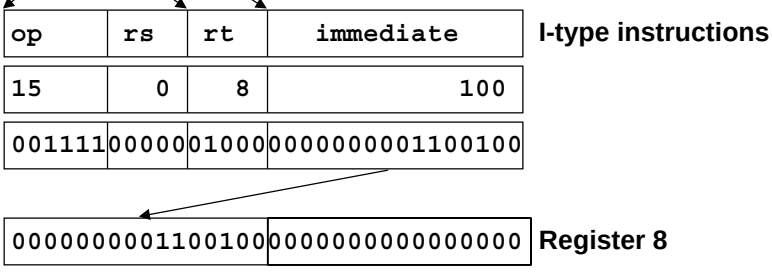


lui, lui

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
```



I-type instructions

Register 8