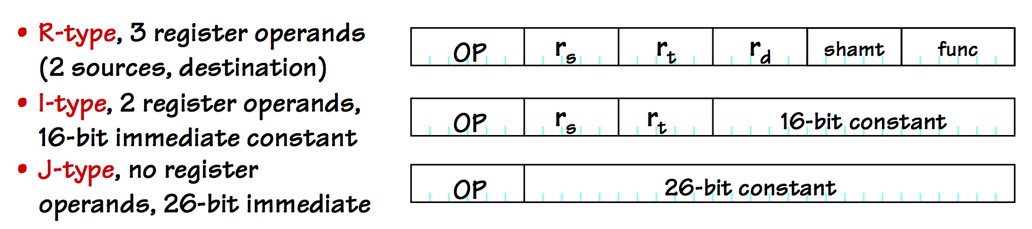
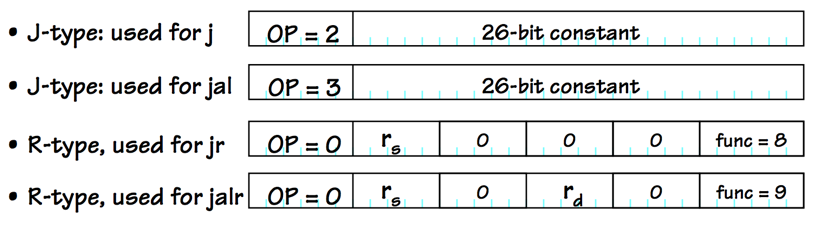
# Lecture 2: Information Theory (Aug 20)

* 1. Number of bits: **log2(N/M) bit=s of information** with N total probable choices and M choices of getting it right
  2. Average information given is **Entropy**
  3. Variable length encoding and decoding

# 2.) Lecture 3 (Aug 25)

1. Signed numbers
2. 2’s Compliment
3. **Negation** = invert every bit and then add 1
4. Fixed point numbers (**for fractions/decimals**)
   1. Cannot describe irrational numbers (i.e. 1/3)
5. **IEEE 754 Format**
6. **Bias notation**
7. Overflow – when arithmetic breaks breaks the contract of 2’s compliment max size
8. Computer organization ( I/O 🡨🡪 CPU 🡨🡪 Memory)

# 3.) Lecture 4-5 (Aug 27-Sept 1)

1. Anatomy of assembly language
2. Anatomy of Von Neumann Computer
3. 8-bit chunks (bytes), 16-bit (shorts), 32-bit (words), 64-bits (words/doubles)
4. 
5. ADD vs ADDU vs ADDI
   1. addu does not generate exception on overflow
   2. add generates an exception on overflow
   3. addi adds an unassigned constant
6. constants in registers
7. MIPS jumps:
   1. 
8. Table describing all MIPS operations

# 4.) Lecture 6 (Sept. 3)

1. **One-, two- and multi-address machines**
2. Arrays in MIPS

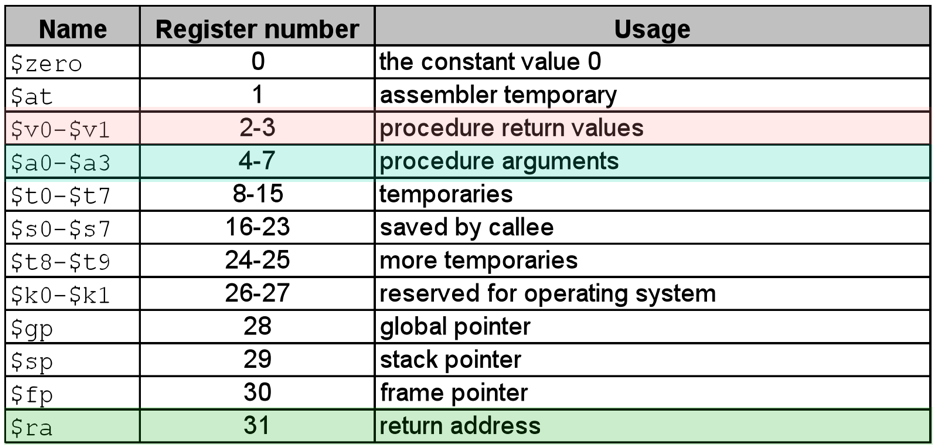
# 5.) Lecture 7 (Sept. 8)

1. Compilers and Interpreters
2. Pointers

# 6.) Lecture 8 (Sept. 10)

1. Assemblers and Linkers
2. How assemblers works
3. Allocation of data storage (Register $sp)

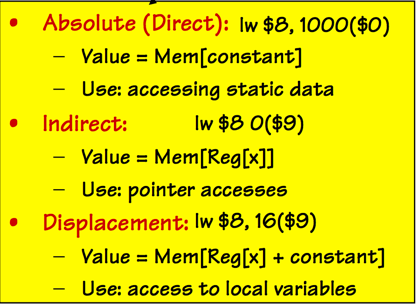
# 7.) Lecture 9 (Sept. 15)

1. Psudoinstructions
2. Procedures
   1. Caller
   2. Callee
3. 
4. Recursion – a callee who call’s itself
5. jr is *return*
6. Dynamic Storage – Stacks & Stack Frames
7. Code lawyer – caller and callee contracts

# 8.) Lecture 10 (Sept. 17)

1. Recursive assembly examples

# Extras

* BEQ multiplies the immediate by 4
* LW treats operand as a signed value
* 
* Compilers normally use addu