Lecture 25

RISC-V Assembly Instructions

- Unconditional branch instructions (aka jumps always)

Example

- Decrement s8 by 1, loop back if s8 is nonzero

```
Loop1: addi s8, s8, -1
bnez s8, Loop1
```

- Increment s8 by 1, loop back if s8 is equal to s9

```
Loop2: addi s8, s8, 1
beq s8, s9, Loop2
```

- Translate C code

```
if (s8 > s9){
    // THEN code
} else{
    // ELSE code
}
// AFTER code
```

- In assembly,

```
ble s8, s9, ELSE1
THEN1: #some code
j AFTER1
ELSE1: #some else code
AFTER1: #some after code
```

- Example 2

```
if(s8 === 5){
    // THEN code
} else{
    // ELSE code
}
// AFTER code
```

- In assembly,

```
addi s9, zero, 5
beq s8, s9, THEN2
ELSE2: #else code
j AFTER2
THEN2: #then code
AFTER2: #after code
```

- Example 3

```
for (s8 = 1; s8 < 5; s8++){
    s9 = s9 + s10;
}
```

- In assembly

```
addi s8, zero, 1
addi t0, zero, 5
Loop3: bge s8, t0, DONE
add s9, s9, s10
addi s8, s8, 1
j Loop3
DONE: #some code
```

- Example 4

```
while (s8 > 8){
    s9 = s9 + s10;
    s8--;
}
```

- In assembly

```
addi t0, zero, 8
Loop4: ble s8, t0, DONE2
add s9, s9, s10
addi s8, s8, -1
j Loop4
DONE2: #some code
```

Machine Code

- Assembly is human readable
- Circuits only taks 1's and 0's
- RISC-V uses 32-bit instructions
- 4 main instruction formats
  - R type (register)
  - I type (immediate)
  - S/B type (store/branch)
  - U/J type (upper immediate/jump)
- Converting to machine code by hand is just tedious
- R type: funct7, rs2, rs1, funct3, rd, op
  - `add s2, s3, s4`
  - op code of 51, funct7 = funct3 = 0, s2 is x18, s3 is x19, s4 is x20
  - = 0b00000001010010011000100100110011 = 0x01498933
- I type: Imm12, rs1, funct3, rd, op
  - `addi s0, s1, 15`

- funct3 = 0, op = 19, s0 = x8, x1 = x9
- = 0b00000000111101001000010000010011 = 0x00F48913