Lecture 26

Example

```
addi s0, zero, 0
addi s1, zero, 4
addi s2, zero, 0
LOOP: BEQ s1, s0, END
add s2, s2, s1
addi s1, s1, -1
j LOOP
END: ebreak
```

The value in s2 after the last line has executed is 10.

Subroutines in Assembly

- Makes code modular, easy to reuse
- Input arguments, return values

Branching to Subroutines

- Not exactly a regular branch
- Has to return to instructions after calling
- use jump and link: jal LABEL, saves address of insruction after it into ra

Example

```
int main() {
    simple();
}
void simple() {
    return;
}
```

In assembly,

```
mem addr LABEL instruction
0x00000300 main: jal simple
0x00000304 #next instruction
0x0000051C simple: jr ra
```

Example Program (Lab 7)

- Search for longest string of 1's in a word of data
- data = 0001100011110011
- shift by 1 then AND the result, count the number of times until the result becomes zero

```
.data
LIST: .word 0x103FED0F
.text
.global _start
_start: la s2, LIST
lw s3, 0(s2)
addi s4, zero 0
LOOP: beqz s3, END #loop until s3 = 0
srli s2, s3, 1
and s3, s3, s2
addi s4, s4, 1 #count
j LOOP
END: ebreak
```

Example: using a subroutine to processs a list of data words

```
LIST: .word 0x0, 0xABCD, 0xFF0011FF, -1
# turn above code into subroutine called ONES
# LOOP over words in LIST and call ONES for each word
```

Passing Arguments and Returning Values

- Calling conventions
  - must agree where to put arguments, return value
  - callee must not interfere with behaviour of caller
- Arguments
  - use registers `a0` to `a7`
  - left to right as needed
  - use stack if more arguments are needed
- Return value: `a0`

Example

```
int main(){
    return add6(11,22,33,44,55,66);
}
int add6(int a, int b, int c, int d, int e, int f){
    return a + b + c + d + e + f;
}
```

In assembly,

```
_start: addi a0, zero, 11
    addi a1, zero, 22
    addi a2, zero, 33
    addi a3, zero, 44
    addi a4, zero, 55
    addi a5, zero, 66
```

```
    jal add6
END: ebreak
add6:   add s1, a0, a1
    add s2, a2, a3
    add s3, a4, a5
    add s1, s1, s2
    add s1, s1, s3
    add a0, s1, zero # return goes into a0
    jr ra # ra takes us back to "END: ebreak"
```