Lecture 28

Subroutines

- In C

```
int FINDSUM(int n){
    int sum = 0;
    while (N!=0){
        sum = sum + N;
        N = N-1;
    }
    return sum;
}
```

- In Assembly

```
.data
.text
.global _start
_start: la s4, N
    lw a0, 0(s4) #a0 = 5
    jal findsum
END: ebreak
findsum: addi t0, zer0, 0
while:  beqz a0, ENDLOOP
    addi a0, a0, -1
    j while
ENDLOOP:add a0, t0, zero
    jr ra
```

- Recursive in C

```
int FINDSUM(int n){
    if(N==0){
        return 0;
    }
    else{
        return FINDSUM(n-1);
    }
}
```

- In assembly

```
.text
.global _start
_start: lw a0, 0(s4)
    jal FINDSUM
END: ebreak
FINDSUM:bnez a0, PUSH
```

```
    jr ra
PUSH:   addi sp, sp, -8
    sw a0, 4(sp)
    sw ra, 0(sp)
    addi a0, a0, -1
    jal FINDSUM
EEE:    lw t0, 4(sp)
    add a0, a0, t0
    lw ra, 0(sp)
    addi sp, sp, 8
    jr ra
```

Wrap Up

- Subroutines make your code modular
- Processor needs to be able to remember where subroutine was called from
- Stack can be used to pass arguments and save/restore registers used by a subroutine