Recitation 12/4

Caller-save versus callee-save

– **Caller Save:** registers freely usable by the callee function

- If needed, caller function saves values on the stack before invoking the callee function
- Callee doesn’t have to worry about overwriting important information needed by the caller

– **Callee Save:** registers that must be restored by the callee function

- Save registers’ values on the stack in the prologue
- Restoration in the epilogue

Which are Which? Can you see a pattern?

Caller Save:

x1 (ra)

x5 (temp or alt link register)

x6 (t1)

x7 (t2)

x10, x11 (first two args and return values)

x12 - x17 (other 6 args)

x28 - x31 (temp registers)

Callee Save:

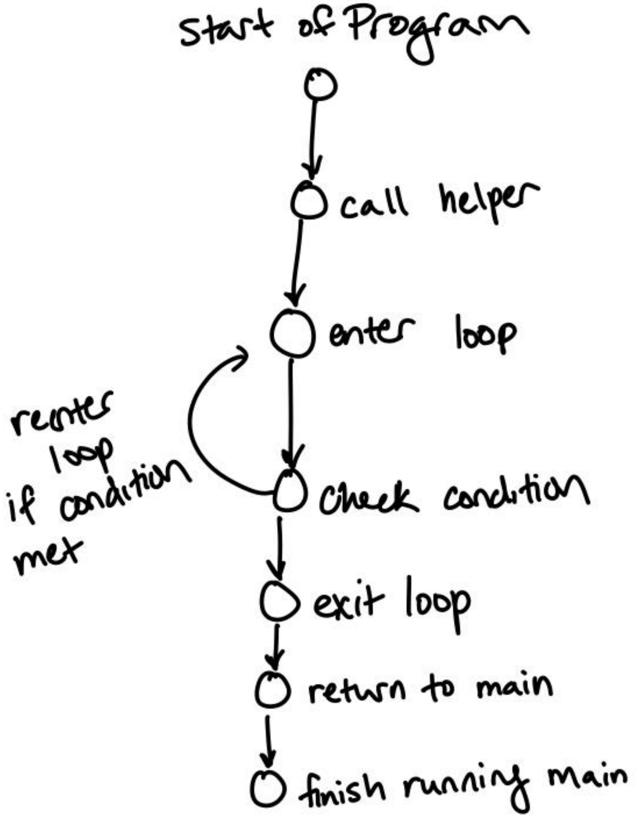
x2 (sp)

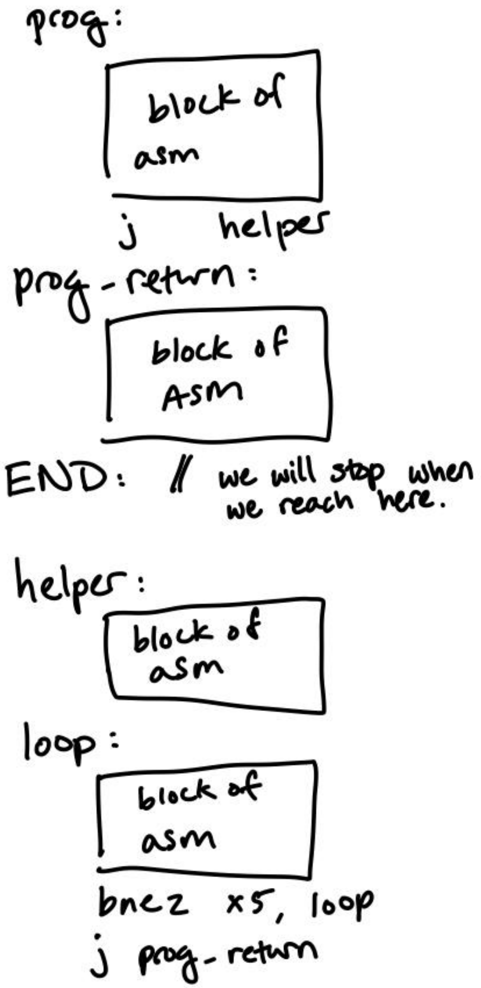
x8 (fp)

x9, x18 - x27 (save registers)

Blocks of Code

- Take advantage of labels!
- When writing the j compiler, sometimes helpful to group instructions together -> one block of code
- Then you can make a control-flow diagram





```
prog:
    block of asm
j helper
prog-return:
    block of ASM
END: // we will stop when
        we reach here.

helper:
    block of asm
loop:
    block of asm
bnez x5, loop
j prog-return
```

2 Cases in ASM:

```
Label1:
        //instr1
        //instr2
        //instr3
        //no instr above were jumps or branches
Label2:
        //instr4
        //instr5
…
```

```
Label1:
        //instr1
        //instr2
        //instr3
        j Label3 //can also be replaced with a branch
Label2:
        //instr4
        //instr5
Label3:
        //more code here
```

Frame and Frame Pointers

- Distinction between frame pointer (register) and frame pointer (an address)
- x8 holds the current function’s frame pointer