Week 8, part D: Loops

What is This Code?

```
main:    add $t0, $zero, $zero
         addi $t1, $zero, 100
START:   beq $t0, $t1, END
         addi $t0, $t0, 1
         j START
END:
```

Loops in MIPS

- Example of a simple loop, in assembly:

```
main:    add $t0, $zero, $zero
         addi $t1, $zero, 100
START:   beq $t0, $t1, END
         addi $t0, $t0, 1
         j START
END:
```

- …which is the same as saying (in C):

```
int i = 0;
while (i != 100) {
   i++;
}
```

Loops in MIPS

```
for ( <init> ; <cond> ; <update> ) {
    <for body>
}
```

- For loops (such as above) are usually implemented with the following structure:

```
main:    <init>
START:   if (!<cond>) branch to END
         <for-body>
UPDATE:  <update>
         jump to START
END:
```

Loop example in MIPS

```
j = 0;
for ( i=0 ; i!=100 ; i++ ) {
      j = j + i;
 }
```

- This translates to:

```
#  $t0 = i, $t1 = j
main:    add $t0, $zero, $zero        # set i to 0
         add $t1, $zero, $zero        # set j to 0
         addi $t9, $zero, 100         # set $t9 to 100
START:   beq $t0, $t9, EXIT           # branch if i==100
         add $t1, $t1, $t0            # j = j + i
UPDATE:  addi $t0, $t0, 1             # i++
         j START
EXIT:
```

- `while` loops are the same, without the initialization and update sections.

Summary

- Assembly is not sophisticated.
  - You have to tell it to do everything.
  - The difficulty comes from its simplicity.
- Making an assembly program:
  - Split to very basic steps.
  - Understand what variables you'll need and set aside registers.
  - Add labels, branches as needed.
  - Thank previous generations for compilers.

Corrado Böhm

Grace Hopper

Homework

- Fibonacci sequence:
  - How would you convert this into assembly?

```
int n = 10;
int f1 = 1, f2 = 1;
int temp;

while (n != 0) {
    temp = f1;
    f1 = f1 + f2;
    f2 = temp;
    n = n - 1;
}
# result is f1
```