“Hand Compiling” C statements to ASM

The Idea

- C and ASM correspond nearly 1 to 1.
- Every C statement can be used to fill in an associated ASM “template”.
- The resulting ASM will then perform the same computation.

Variables, Temporaries, and Assignment

- Each C (int, pointer) variable or temporary value should map to one register.

```
int a = 5;
int b = 3 * a + 1;
```

- Registers can be reused, either because you ran out or as an optimization.

- $t0 is a
- $t1 is b
- $t2 is 3 (no muli)
- $t3 is (3 * a)

```
li $t0, 5
li $t2, 3
mul $t3, $t2, $t0
addi $t1, $t3, 1
```

Which Registers

You, the programmer, “own” the following registers:

- `$t0 .. $t9`
- `$s0 .. $s7`

You set these registers, and once you’ve set them they should stay that way until you change them.*

(* Technically not true for $t registers…)

All other registers have some specific purpose, and various instructions may set them as an invisible side effect or convention may expect them to contain specific things at specific times.

Put your values into special registers as late as possible, and if you want to keep a value in a special register move it out to a T or S register ASAP.

- `$a0 .. $a3, $v0, $v1`
- `$gp, $sp, $fp, $ra`
- `pc, hi, lo`

if statements

```
// case 1
if (x < y) {
    y = 7;
}


// case 2
if (x < y) {
    y = 7;
}
else {
    y = 9;
}
```

```
# x is $t0
# y is $t1

# case 1
     bge $x0, $x1, skip_label # inverted
     li $x1, 7
done_label:

# case 2
     blt $x0, $x1, then_label:
     j else_label
then_label:
     li $x1, 7
     j done_label
else_label:
     li $x1, 0
done_label:
```

do-while loops

```
do {
    x = x + 1;
} while (x < 10);
```

```
# x is $t0
# 10 is $t1

do_label:
    addi $t0, $t0, 1
    li $t1, 10
    blt $t0, $t1, do_label
```

while loops

```
while (x < 10) {
    x = x + 1;
}
```

```
# x is $t0
# 10 is $t1

while_test:
        li $t1, 10
        bge $t0, $t1, done_label:  # inverted
# while body
        addi $t0, $t0, 1
# }
        j while_test
done_label:
```

complex for loop

```
for (int i = 0; i < 10 && x != 7; ++ii) {
     x = x + 3;
}

==

for (int i = 0;
          i < 10 && (x < 7 || x > 7);
          ++ii) {
  ...
```

```
# x is $t0              i is $t1
# i < 10 is $t2         7 is $t3
# x < 7  is $t4         7 < x  is $t5
# x != 7 is $t6         cond   is $t7

      li $t1, 0
for_test:
      slti $t2, $t1, 10
      li $t3, 7
      slt $t4, $t0, $t3
      slt $t5, $t3, $t0
      or  $t6, $t4, $t5
      and $t7, $t6, $t2
      beq $t7, $zero, for_done
# for body
      addi $t0, $t0, 3
# for inc
      addi $t1, $t1, 1
      j for_test
for_done:
```