Assembly: conditionals; `goto`

COSC 208, Introduction to Computer Systems, 2021-10-15

Announcements

Outline

- Warm-up
- Conditionals
- `goto`

Warm-up

- Q1: *Assume the registers currently hold the following values:*

```
sp = 0xA980
w/x0 = 0
w/x1 = 1
w/x2 = 2
w/x3 = 3
w/x4 = 4
w/x5 = 5
```

Draw the contents of the stack after the following instructions have been executed:

```
sub sp, sp, #0x30
str w0, [sp, #16]
str x1, [sp]
str w2, [sp, #20]
str x3, [sp, #32]
str w4, [sp, #28]
str w5, [sp, #8]
```

```
        0  1  2  3  4  5  6  7  8
    ->  +--+--+--+--+--+--+--+--+
0xA9__  |                       |
        +--+--+--+--+--+--+--+--+
...
```

```
0xA980
```

goto example from last class

More practice

```
1  int flip(int bit) {
2     int result = -1;
3     if (bit == 0) {
4          result = 1;
5      }
6      else {
7          result = 0;
8      }
9      return result;
10 }
```

- Q2: *The above C code was compiled into assembly. Label each line of assembly code with the line number of the line of C code from which the assembly instruction was derived.*

```
0000000000400544 <flip>:
    400544: d10043ff    sub sp, sp, #0x10
    400548: 12800008    mov w8, #0xffffffff
    40054c: b9000fe0    str w0, [sp, #12]
    400550: b9000be8    str w8, [sp, #8]
    400554: b9400fe8    ldr w8, [sp, #12]
    400558: 35000088    cbnz    w8, 400568 <flip+0x24>
    40055c: 52800028    mov w8, #0x1
    400560: b9000be8    str w8, [sp, #8]
    400564: 14000002    b   40056c <flip+0x28>
    400568: b9000bff    str wzr, [sp, #8]
    40056c: b9400be0    ldr w0, [sp, #8]
    400570: 910043ff    add sp, sp, #0x10
    400574: d65f03c0    ret
```

- Q3: *How does the C code and its assembly differ in terms of the conditional execution? i.e. compare and contrast the `else` and the two branches.*

```

```

- Q4: *Write a function called `flip_goto` that behaves the same as `flip` but matches the structure of the assembly code that will be generated for `flip`. (Hint: you'll need two `goto` statements.)*

Practice with conditionals

Write the C if-statement code equivalent for each snippet of assembly, treating registers as if they were variable names.

- Q5:

```
cmp w0, w1
b.eq 0xABCD <foo+0x40>
```

```

```

- Q6:

```
cmp w0, #0x20
b.lt 0xABCD <foo+0x80>
```

```

```

- Q7:

```
cmp w1, #0x1
b.ne 0xABCD <foo+0xC0>
```

```

```

- Q8:

```
cmp w0, w1
b.le 0xABCD <foo+0xF0>
```

```

```