Control Flow in Assembly Language

CSE 102

Assembly Control Flow

- Unconditional Jump –

```
jmp   [label]
```

e.g.,

```
      jmp   L1
            …
L1:   target instruction
            …

ijmp        indirect jump, dest in Z (r31:r30)
```

Conditional Control Flow

- In AVR, separate expression eval and cond branch inst.
- Compare –

```
cp    Rd, Rr
```

- Perform operation temp = Rd – Rr, throw away temp and set flags based on results of subtraction operation.
- SREG flags can also be set as a result of normal arithmetic and/or logical operations.

Conditional Branches

```
br[cond]    [label]
```

e.g.,

```
brge        br_loop     ;branch if greater or
                        ;equal (signed)
```

- There are three classes of conditionals:
  - General
  - Unsigned
  - Signed

General Conditionals

breq	zero (Z set)
brne	not zero (Z cleared)
brcs	carry (C set)
brcc	no carry (C cleared)
brvs	overflow (V set)
Brvc	no overflow (V cleared)

Signed Conditionals

brlt	less than (a < b)
brge	greater or equal (a ≥ b)

Unsigned Conditionals

brlo	branch if lower (a < b)
brsh	branch if same or higher (a ≥ b)

Control Flow in C

```
if … then …                    e.g.,
    if ([cond expr]) {             if ( var1 > var2 ) {
        [true body]                    var1 = var1 + var2;
    }                                  var2 = 0;
    [main body]                    }
                                   …
```

if … then

```
if … then …                    assembly

    if ([cond expr]) {             cp  [cond exp opers]
        [true body]                br[!cond] main_body
    }                              [true body]
    [main body]                main_body:
                                   [main body]
```

if … then … else

```
if ([cond expr]) {             cp          [cond exp opers]
    [true body]                br[!cond]   false_body
}                              [true body]
else {                         jmp         main_body
    [false body]           false_body:
}                              [false body]
[main body]                main_body:
                               [main body]
```

Conditional if … then … else

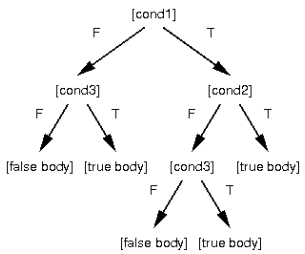
```
if (([cond1] && [cond2]) || [cond3]) {
    [true body]
}
else {
    [false body]
}
[main body]
```

- Note: evaluation order of compound expression is left to right, only conditions that need to be evaluated are evaluated

Conditional if … then … else

if (([cond1] && [cond2]) || [cond3])

Evaluation order for above compound expression:



if (([cond1] && [cond2]) || [cond3])

```
        cp          [cond1]
        br[!cond1]  check_cond3
        cp          [cond2]
        br[cond2]   true_body
check_cond3:
        cp          [cond3]
        br[cond3]   true_body
        [false body]
        jmp         main_body
true_body:
        [true body]
main_body:
        [main body]
```

for loop

```
for ([ind var] = [init val]; [cond expr]; [update ind var] ) {
        [loop body]
}
[main body]
```

e.g.,

```
for (i=0; i<24; i++) {
    mask = 1 << i;
    status_bit[i] = status & mask;
    status_bit[i] >>= i;
}
```

for loop

```
for ([ind var] = [init val]; [cond expr]; [update ind var] ) {
        [loop body]
}
[main body]
```

- assembly

```
            lds         [ind var], [init val]
for_loop:   cp          [cond expr]
            br[!cond]   loop_exit
            [loop body]
            [update ind var]
            jmp         for_loop
loop_exit:
            [main_body]
```

while loop

```
while ([cond expr]) {
        [loop body]
}
[main body]
```

- assembly

```
        while_loop:
                cp          [cond expr oper]
                br[!cond]   exit_while
                [loop body]
                jmp         while_loop
        exit_while:
                [main body]
```