

Control Flow

if ... then

```
if ([cond expr]) {  
    [true body]  
}  
[main body]
```

e.g.

```
if (var1 > var2) {  
    var1 = var1 + var2;  
    var2 = 0;  
}
```

```
cmpl [cond exp operands]  
j [! cond] main_body
```

```
[true body]  
main_body:  
[main body]
```

```
movl var1, %eax  
cmpl var2, %eax  
jbe main_body  
addl var2, %eax  
movl %eax, var1  
movl $0, var2  
main_body:  
;
```

if then else

```
if ([cond expr]) {  
    [true body]  
} else {  
    [false body]  
}  
[main body]
```

```
cmpl [operands]  
j [! cond] false_body
```

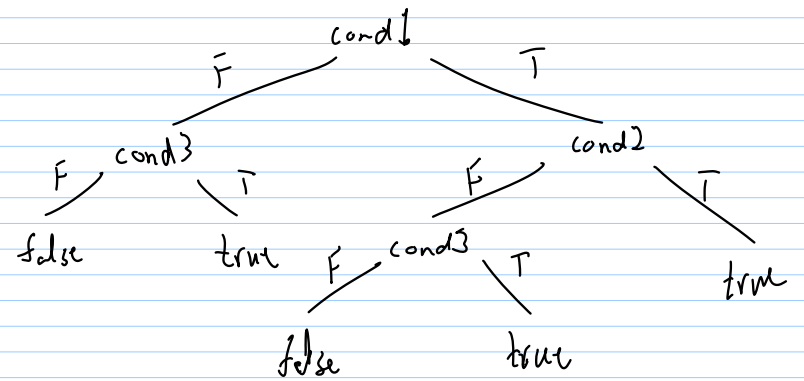
```
[true body]  
jmp main_body  
false_body:  
[false body]  
main_body:  
[main body]
```

compound if then else

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

eval order is left to right, only what is req. to determine result

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



```
comp1 [cond1]  
j[! cond1] check_cond3  
comp2 [cond2]  
j[cond2] true_body  
check_cond3:  
    comp3 [cond3]  
    j[cond3] true_body  
    [false body]  
true_body: jmp main_body  
main_body: [main body]
```

for loop

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

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

00010000 ← i

```

movl [init_val], [ind var]
for_loop:
    cmpl [cond]
    j [! cond] loop_exit
    [loop body]
    [update ind var]
    jmp for_loop
loop_exit:
    [main body]

```

```

while loop
while ([cond]) {
    [loop]
}

```

```

while-loop:
    cmpl [cond]
    j [! cond] exit_while
    [loop body]
    jmp while-loop
exit_while:

```

Gotchas for Java programmers

```

int i;
for (i=0; i<10; i++) ...

```

NOT

```

for (int i; i<10; i++){...

```

Uninitialized variables

Error Handling

```

int main () {
    int i;

```

```

    factorial(i);
    :
}

```

in C, no exceptions

convention - return values
 0 return success
 ≠ 0 return failure