

# Control Flow

if ... then

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

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

```

    cmpd    [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 exp] ) {  
    [true body]  
} else {  
    [false body]  
}  
[main body]
```

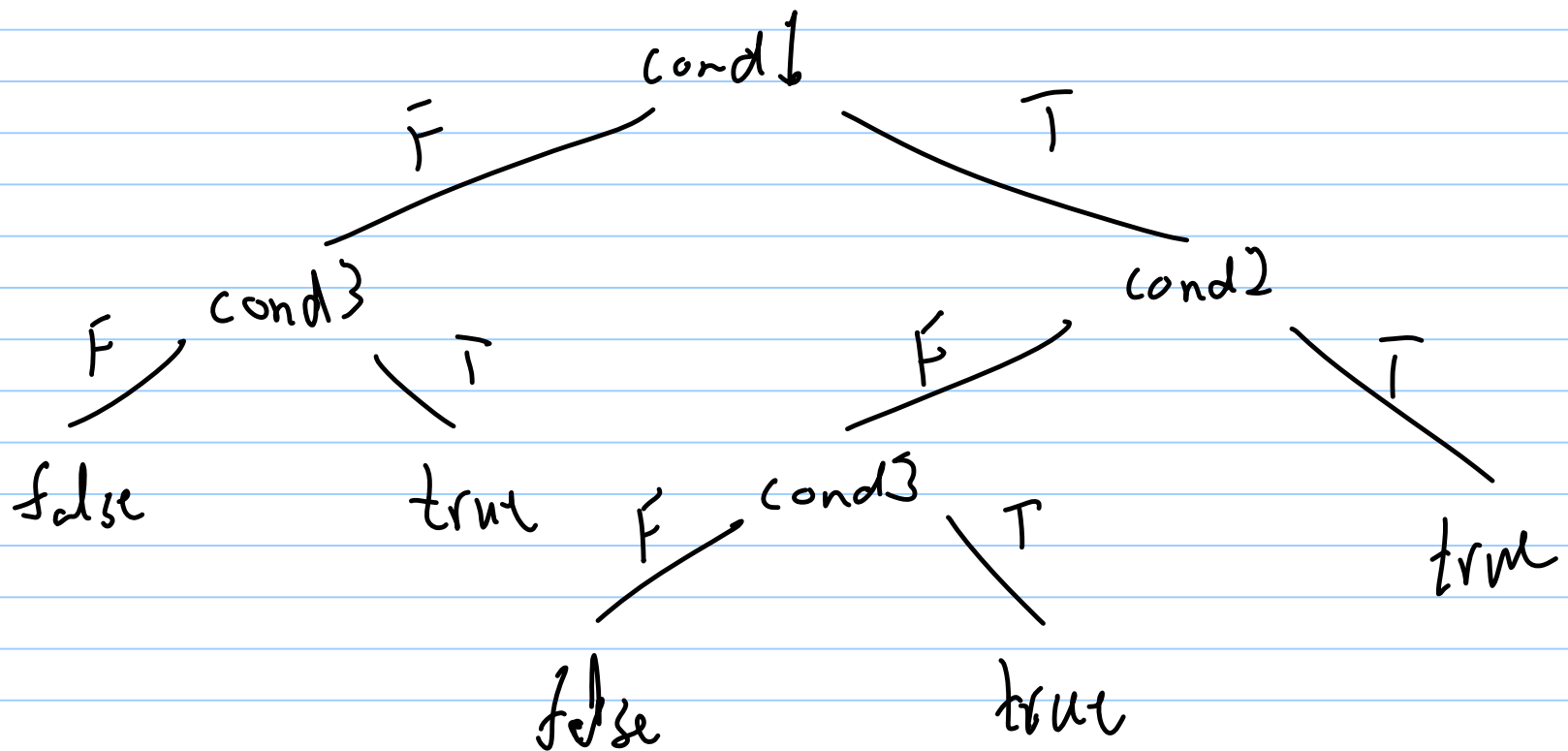
compd [operands]  
j[! cond] false\_body  
[true body]  
jmp main\_body  
false\_body:  
[false body]  
main\_body:  
[main body]

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)



```

      cmpl [cond1]
      j[! cond1] check_cond3
      |
      cmpl [cond2]
      j[cond2] true_body
check_cond3:
      cmpl [cond3]
      j[cond3] true_body
      [false body]
true_body:  jmp main_body
            [true body]
main_body: [main body]

```

```
for loop
```

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

```
for (i = 0; i < 24; i++) {
```

```
    mask = 1 << i;
```

```
    status-bit[i] = status & mask;
```

```
}
```

000 | 0000  
          ← i

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

while loop

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

while-loop:

```
    compl [cond]  
    if [! 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

```
int main () {  
    int i;  
  
    factorial(i);  
    :  
}
```

Error Handling

in C, no exceptions

convention - return values  
0 return success  
≠ 0 return failure