Assembly Control Flow

Friday, October 23, 2020 12:43 PM

Assembly Execution

-> Program vals in many registers
-> Asm instructions stored in memory

Program Counter (PC) %rip special register
stores address of next
instruction to execute

Update using
%rip += size of bytes of current instruction

Unconditional Looping

-> Interfere with %rip, set it back to earlier instruction

jmp moves %rip

jmp Label direct jump
jmp *Operand indirect jump

ex.

```
jmp  404f8   <loop + 0x6>   # instruction at 404f8
jmp  *%rax                  # instruction at address in %rax
```

Control

C

```
if (x > y) {
    //a
} else {
    //b
}
```

asm

Calculate condition, then execute code

Compare `cmp S1, S2`

Execute

```
je/jz    # jump if equal
jne/jnz  # jump if not equal
jl/jge   # jump if less
:
```

ex.

```
cmp $2, %edi
jg [target]    //jump if %edi > 2
```

CPU has condition code ("global vars") automatically keeping track of info about most recent arith, logical operations - cmp stores, conditional jmp reads this

(Single-bit registers)

ex.

CF Carry flag - detect MSB carry out (unsigned overflow)

ZF Zero flag - yielded zero

SF Sign flag - yielded negative

OF Overflow flag - two's complement overflow

⋮

cmp does subtraction, updating condition codes

test does same thing but with bitwise AND

Other arith/logical instructions also update condition codes except lea

If statements

C

```
int if_then(int param1) {
    if (param1 == 6) {
        param1 += 1;
    }
    return param1 * 2;
}
```

asm

```
d6  cmp  $0x6, %edi
d9  jne  de
dd  addl $0x1, %edi
de  lea  (%rdi, %rdi, 1) %eax
e1  retq
```

asm pseudocode

Test
Jump to else-body if test fails
If-body
Jump to past else-body
Else-body
Past else-body

Loops

While loops

Jump to test
Body
Test
Jump to body if success

For loop

Init
Jump to test
Body
Update
Test
Jump to body if success

Condition code instructions other than jmp

set conditionally sets byte to 0 or 1

mov new versions can move if condition true