CS 33
Intro to Machine Programming
Machine Model

Processor (aka CPU)  Memory (aka RAM)

instructions and data

data
Memory

Instructions or Instructions are Data

Data
Processor: Some Details

- Execution engine
- Instruction pointer
- Condition codes
Processor: Basic Operation

```c
while (forever) {
    fetch instruction IP points at
    decode instruction
    fetch operands
    execute
    store results
    update IP and condition code
}
```
Instructions ...

<table>
<thead>
<tr>
<th>Op code</th>
<th>Operand1</th>
<th>Operand2</th>
<th>...</th>
</tr>
</thead>
</table>
Operands

• Form
  – immediate vs. reference
    » value vs. address

• How many?
  – 3
    » add a,b,c
      • c = a + b
  – 2
    » add a,b
      • b += a
Operands (continued)

• Accumulator
  – special memory in the processor
    » known as a register
    » fast access
  – allows single-operand instructions
    » add a
      • acc += a
    » add b
      • acc += b
From C to Assembler ...

\[ a = (b + c) \times d; \]
\[ \text{if} \ (a < b) \]
\[ c = 1; \]
\[ \text{else} \]
\[ d = 1; \]
\[ \text{cmp} \ a, b \]
\[ \text{jge} \ \text{.L1} \]
\[ \text{mov} \ \$1, c \]
\[ \text{jmp} \ \text{.L2} \]
\[ \text{.L1} \]
\[ \text{mov} \ \$1, d \]
\[ \text{.L2} \]
Condition Codes

• Set of flags including status of most recent operation:
  – zero flag
    » result was or was not zero
  – sign flag
    » result was or was not negative (sign bit is or is not set)
  – overflow flag
    » for values treated as signed
  – carry flag
    » for values treated as unsigned

• Set implicitly by arithmetic instructions
• Set explicitly by compare instruction
  – cmp a,b
    » sets flags based on result of b-a
Jump Instructions

• Unconditional jump
  – just do it

• Conditional jump
  – to jump or not to jump determined by condition-code flags
  – field in the op code indicates how this is computed
  – in assembler language, simply say
    » je
      • jump on equal
    » jne
      • jump on not equal
    » jgt
      • jump on greater than
    » etc.
Addresses

```c
int a, b, c, d;

int main() {
    a = (b + c) * d;
    ...
}
```

```
mov b, %acc
add c, %acc
mul d, %acc
mov %acc, a
mov 1004, %acc
add 1008, %acc
mul 1012, %acc
mov %acc, 1000
```
Addresses

```c
int b;

int func(int c, int d) {
    int a;
    a = (b + c) * d;
    ...
}
```

- One copy of $b$ for duration of program’s execution
  - $b$’s address is the same for each call to `func`
- Different copies of $a$, $c$, and $d$ for each call to `func`
  - addresses are different in each call

```assembly
mov  ?,%acc
add  ?,%acc
mul  ?,%acc
mov  %acc,?
```
Relative Addresses

- **Absolute address**
  - actual location in memory

- **Relative address**
  - offset from some other location

- **Blob's absolute address is 10000**
- **Datum’s relative address (to Blob) is 100**
  - its absolute address is 10100
Base Registers

```assembly
mov $10000, %base
mov $10, 100(%base)
```
Addresses

```c
int b;

int func(int c, int d) {
    int a;
    a = (b + c) * d;
    ...
}
```

```
mov 1000,%acc
add c_rel(%base),%acc
mul d_rel(%base),%acc
mov %acc,a_rel(%base)
```
Quiz

Suppose the value in base is 10,000 and c_rel is -8. What is the address of c?

a) 9992  
b) 9996  
c) 10,004  
d) 10,008

mov 1000,%acc  
add c_rel(%base),%acc  
mul d_rel(%base),%acc  
mov %acc,a_rel(%base)

Suppose the value in base is 10,000 and c_rel is -8. What is the address of c?

a) 9992  
b) 9996  
c) 10,004  
d) 10,008
Registers

- Instruction pointer
- Accumulator
- Base register
- More
- Condition codes

Executable engine

Interchangeable