Machine Prog: Basics

李舒辰

2019 年 9 月 19 日

Outline

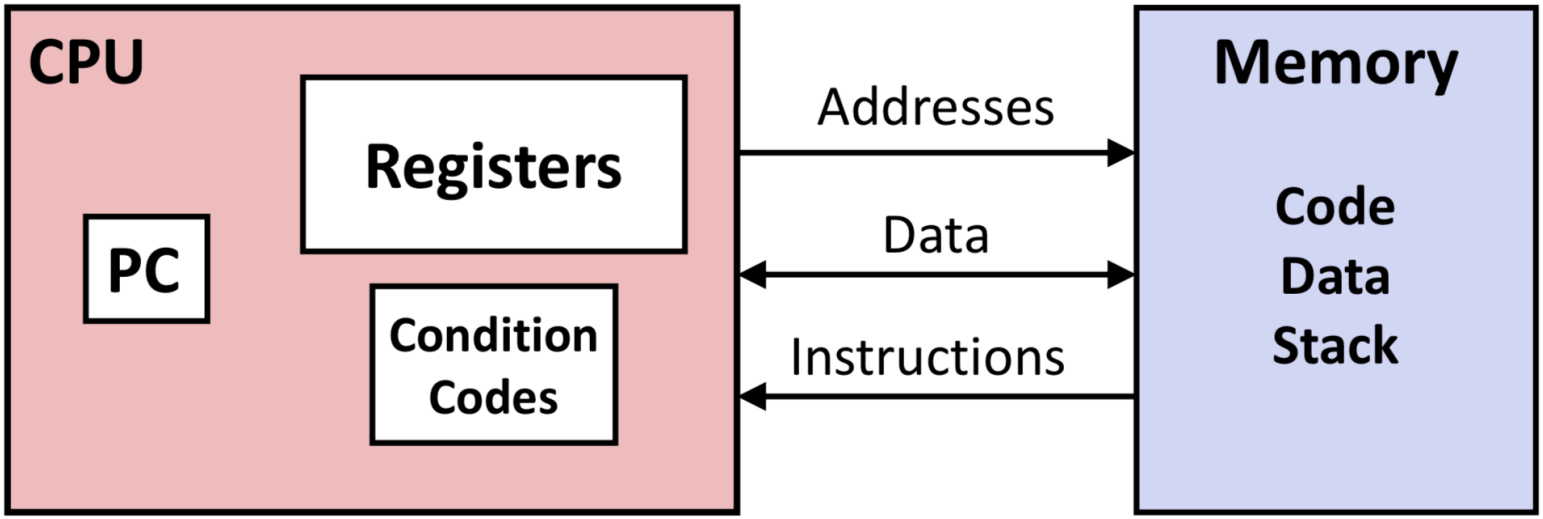
Assembly and Machine Code

Terminology

Terminology

- Architecture (instruction set architecture)
  - Code Forms
    - Machine Code
    - Assembly Code
- Microarchitecture

Interactions between CPU and Memory



C Code → Object Code

Compile: `gcc -Og *.c -o *`

C Code → Object Code

Compile: `gcc -Og *.c -o *`

```
$ gcc -O
fast  -- optimize for speed disregarding exact standards compliance
g     -- optimize for debugging experience
s     -- optimize for space
0  1  2  3
```

C Code → Object Code

`*.c`

C Code → Object Code

```
*.c
  ↓   gcc -Og -S *.c -o *.s
*.s
```

C Code → Object Code

```
*.c
  ↓   gcc -Og -S *.c -o *.s
*.s
  ↓   gcc -c *.s -o *.o
*.o
```

C Code → Object Code

```
*.c
  ↓   gcc -Og -S *.c -o *.s
*.s
  ↓   gcc -c *.s -o *.o
*.o
  ↓   gcc *.o
  *
```

Assembly Characteristics: Data Types

Assembly Characteristics: Data Types

- Integral data
  - values
  - addresses
- Floating point data
- Instructions

Assembly Characteristics: Operations

Assembly Characteristics: Operations

- Perform arithmetic function
- Transfer data
- Transfer control

Disassembling

Disassemble: `objdump -d *`

Disassembling

Disassemble: `objdump -d *`

- Display information from object files.

Disassembling in gdb

Disassembling in gdb

1. Compile: `$ gcc -g *.c -o *`

Disassembling in gdb

1. Compile: `$ gcc -g *.c -o *`
2. Debug: `$ gdb *`

Disassembling in gdb

1. Compile: `$ gcc -g *.c -o *`
2. Debug: `$ gdb *`
3. Disassemble: `(gdb) disassemble ...`

Registers, Operands, Moving

x86-64 Interger Registers

%rax	%eax
%rbx	%ebx
%rcx	%ecx
%rdx	%edx
%rsi	%esi
%rdi	%edi
%rsp	%esp
%rbp	%ebp

%r8	%r8d
%r9	%r9d
%r10	%r10d
%r11	%r11d
%r12	%r12d
%r13	%r13d
%r14	%r14d
%r15	%r15d

Operand Types

- Immediate: `$-577`, `$0x1f`
- Register: `%rax`, `%ebx`, `%r11`
- Memory: `(%rax)`
  - `D(r_b, r_i, s)` := `Mem[D + Reg[r_b] + Reg[r_i] * s]`

Operand Types

- Immediate: `$-577`, `$0x1f`
- Register: `%rax`, `%ebx`, `%r11`
- Memory: `(%rax)`
  - `D(r_b, r_i, s)` := `Mem[D + Reg[r_b] + Reg[r_i] * s]`

  (`r_i` $\neq$ `%rsp`)

Moving Data

`movq` Src, Dest

Moving Data

`movq` Src, Dest

- Source: Imm, Reg, Mem
- Destination: Reg, Mem

Moving Data

`movq` Src, Dest

- Source: Imm, Reg, Mem
- Destination: Reg, Mem

Cannot have both operands refer to memory locations

Moving Data

```
1 void fun(int *p, int *q) {
2     *p = *q;
3 }
N… mov.c                                  33% ≡     1:  1

1 _fun:
2     movl    (%rsi), %eax
3     movl    %eax, (%rdi)
4     ret
N… mov.s                                  25% ≡     1:  1

[0] 0:vim*                              "mbp" 21:13 18-Sep-19
```

An Example: Swap

An Example: Swap

```
1 void swap(long *xp, long *yp) {
2     long t0 = *xp;
3     long t1 = *yp;
4     *xp = t1;
5     *yp = t0;
6 }
N… swap.c                                   c   14% ≡    1:  1

1 _swap:
2     movq    (%rdi), %rax
3     movq    (%rsi), %rdx
4     movq    %rdx, (%rdi)
5     movq    %rax, (%rsi)
6     ret
N… swap.s                                 asm   14% ≡    1:  1

[3] 0:vim*                                "mbp" 20:29 18-Sep-19
```