10.11

Outline

- Previous homework
- Registers
- Instructions

Homework

```
float_bits float_abs(float_bits f)
{
    float a = *(float*)(&f);
    return a == a ? f & 0x7fffffff : f;
}
```

Registers -- Name

Segment Registers:
CS: code segment; DS: data segment;
ES: extra segment; SS: stack segment

- Origin
  - EAX: The Accumulator
  - EBX: The Base Register
  - ECX: The Counter
  - EDX: The Data Register
  - EDI: The Destination
  - ESI: The Source
  - ESP: Stack Pointer
  - EBP: Base Pointer

- Some instructions to illustrate
- IMULL r32 – EDX:EAX = EAX * r32
  - EDX as data register
- XLATB – AL = [DS:BX + AL]
  - BX: base position of your table
- LOOP label – use ECX/CX as a counter
- MOVSW – move [DS:SI] to [ES:DI]
  - SI: source index, DI: destination index

Current Register Usage

- EAX: passing return values
- EDI, ESI: passing parameters when calling a function
  - Stack is also used for parameter passing

- Caller and callee saved registers

CISC and RISC

- x86 is CISC
  - Many uncommon instructions:
    - Rep, string operation …
- Common RISC architecture: ARM, RISC-V
  - Less instructions, which means smaller binary code
  - More registers – operations always between registers
    - Cannot add value between memory and register

Instructions – arith and logic

- Argument order!
  - Always $2 op= $1 (include mov)
  - CMP instruction: $2 - $1
- Condition codes
  - CF OF ZF SF
  - SET intstruction

Instructions – JMP

- Forget about if-else, for, while and do-while
- Only have branches and loops
  - Change your mind!
- Jumping table for switch-case
  - Faster than if-else
  - Sparse switch-case may be translated into branches instead of jumping table

1. 在下列指令中，其执行会影响条件码中的 CF 位的是：

A. jmp NEXT　　B. jc NEXT　　C. inc %bx　　D. shl $1,%ax

2. 下列关于比较指令 CMP 说法中，正确的是：

A. 专用于有符号数比较　　B. 专用于无符号数比较

C. 专用于串比较　　D. 不区分比较的对象是有符号数还是无符号数

3. 在如下代码段的跳转指令中，目的地址是：

```
400020: 74 F0     je _________
400022: 5d        pop %rbp
```

A. 400010　　B. 400012　　C. 400110　　D. 400112

6. 在如下 switch 语句对应的跳转表中，哪些标号没有出现在分支中？

```
addq $1, %rdi
cmpq $8, %rdi
ja .L2
jmp *.L4(, %rdi, 8)
.L4:    .quad .L9 .quad .L5 .quad .L6 .quad .L7 .quad .L2
        .quad.L7      .quad .L8     .quad .L2     .quad .L5
```

A. 3, 6　　B. -1, 4　　C. 0, 7　　D. 2, 4

8. 假设某条 C 语言 switch 语句编译后产生了如下的汇编代码及跳转表：

```
  movl 8(%ebp), %eax
  subl $48, %eax
  cmpl $8, %eax
  ja .L2
  jmp *.L7(, %eax, 4)
```

```
.L7:
.long .L3
.long .L2
.long .L2
.long .L5
.long .L4
.long .L5
.long .L6
.long .L2
.long .L3
```

在源程序中，下面的哪些（个）标号出现过：

A. ‘2’, ‘7’

B. 1

C. ‘3’

D. 5