ICS Seminar Week3 Prep

王善上 贾博暄 倪嘉怡 许珈铭

2023.9.23

Rules

remainder <- ordinal number in WeChat Group % 4

for all questions do

    if question number % 4 == remainder then

        you should work on it

    end

end

Q1

1. 判断下列 x86-64 ATT 操作数格式是否合法。

(1) (　　) 8(%rax, ,2)

(2) (　　) $30(%rax,%rax,2)

(3) (　　) 0x30

(4) (　　) 13(,%rdi,4)

(5) (　　) (%rsi,%rdi,6)

(6) (　　) %ecx

(7) (　　) (%ecx)

(8)★(　　) (%rbp,%rsp)

1. **F** 有scaling factor但index缺失
2. **F** 这里的Offset不加$
3. **T** 内存地址0x30
4. **T**
5. **F** Scaling factor只能是1,2,4,8
6. **T**
7. **F** 约定：
   x86-64 不允许将除 64 位寄存器以外的寄存器作为寻址模式基地址
8. **F** %rsp 不能作为操作数
   （参考 Intel 手册 Vol.1 3-23 与 Vol.2A 2-7）

F
F
T
T
F
T
F
F

Q2

7、x86体系结构的内存寻址方式有多种格式，请问下列哪些指令是正确的：（　　）

A. movl　$34,　(%eax)

B. movl　(%eax),　%eax

C. movl　$23,　10(%edx, %eax)

D. movl　(%eax),　8(%ebx)

ABC

D 错误，mov不可以从Mem到Mem

注意此题没有限定x86-64。
32位系统中，(%eax)是正确的

ABC

Q3

3．下列操作不等价的是（　　）

A．movzbq 和 movzbl

B．movzwq 和 movzwl

C．movl 和 movslq

D．movslq %eax, %rax 和 cltq

A movzbl会把高位4 bytes设为零，效果同movzbq

B 与A同理

C movl：高位设为零

movslq：符号扩展

D 效果完全相同：

“It therefore has the exact same effect as the instruction movslq %eax, %rax, but it has a more compact encoding”

C

Q4

4. 判断下列 x86-64 ATT 数据传送指令是否合法。

(1) (　　) movl $0x400010, $0x800010

(2) (　　) movl $0x400010, 0x800010

(3) (　　) movl 0x400010, 0x800010

(4) (　　) movq $-4, (%rsp)

(5) (　　) movq $0x123456789AB, %rax

(6) (　　) movabsq $0x123456789AB,%rdi

(7)★(　　) movabsq $0x123456789AB,16(%rcx)

(8)★(　　) movq 8(%rsp),%rip

1. **F** Imm -> Imm 不允许
2. **T** Imm -> Mem
3. **F** Mem -> Mem 不允许
4. **T** 将$-4写入栈顶
5. **F** movq vs movabsq问题
   这里要用movabsq
6. **T**
7. **F**
   movabsq的目标地址必须是整数寄存器(Imm -> Reg)
8. **F**
   不能用mov向%rip中传入数据

F
T
F
T
F
T
F
F

Q5

4. 在 x86-64 下，以下哪个选项的说法是错误的？

A) movl 指令以寄存器作为目的时，会将该寄存器的高位 4 字节设置为 0

B) cltq 指令的作用是将%eax 符号扩展到%rax

C) movabsq 指令只能以寄存器作为目的

D) movswq 指令的作用是将零扩展的字传送到四字节目的

D

s = sign-extend

z = zero-extend

movswq: 符号扩展

D

Q6

(　　)9. 在32位机器下, 假设有如下定义int array[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; 某一时刻, %ecx存着第一个元素的地址, %ebx值为3, 那么下列操作中____将array[3]移入了%eax.

A. leal 12(%ecx), %eax
B. leal (%ecx, %ebx, 4), %eax
C. movl (%ecx, %ebx, 4), %eax
D. movl 8(%ecx, %ebx, 2), %eax

访问内存，用mov而非lea
求Scaling factor:
sizeof(int) = 4
从array[0]到array[3]: 3个sizeof(int)
%ebx值: 3
所以
s = (4 * 3) / 3 = 4
C

(%ecx, %ebx, 4)
= 第一个元素的地址 + 3 * 4

C

Q7

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

D

CF：进位；无符号溢出

AB. jump指令不改变条件码

C. inc, dec：不改变进位标志

D.逻辑运算：进位 置 0

D

Q8

6. X86-64 指令提供了一组条件码寄存器；其中 ZF 为零标志，ZF=1 表示最近的操作得出的结构为 0；SF 为符号标志，SF=1 表示最近的操作得出的结果为负数；OF 为溢出标志，OF=1 表示最近的操作导致一个补码溢出（正溢出或负溢出）。当我们在一条 cmpq 指令后使用条件跳转指令 jg 时，那么发生跳转等价于以下哪一个表达式的结果为 1？

A. ~(SF ^ OF) & ~ZF

B. ~(SF ^ OF)

C. SF ^ OF

D. (SF ^ OF) | ZF

A

以cmp B, A; jg …为例，跳转jg需要满足A - B > 0

1) A – B != 0: ~ZF
2) OF == 0 无溢出: SF == 0 结果为正
3) OF == 1 溢出: SF == 1 结果为负
4) 综合2,3，~ (SF ^ OF) 先异或再取反，再与1取交集

A

Q9

2、条件码描述了最近一次算术或逻辑操作的属性。下列关于条件码的叙述中，哪一个是不正确的？

A. set 指令可以根据条件码的组合将一个字节设置为 0 或 1

B. cmp 指令和 test 指令可以设置条件码但不更改目的寄存器

C. leaq 指令可以设置条件码 CF 和 OF

D. 除无条件跳转指令 jmp 外，其他跳转指令都是根据条件码的某种组合跳转到标号指示的位置

C

lea不改变条件码

C

Q10

5. 将 AX 清零，下列指令错误的是（ ）

A. sub %ax, %ax　　B. xor %ax, %ax

C. test %ax, %ax　　D. and $0, %ax

C

A. %ax -= %ax; -> 0

B. %ax ^= %ax; -> 0

C. %ax &= %ax; -> 1 (if %ax!=0); 0 (if %ax==0)

D. %ax &= 0; -> 0

C

Q11

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

D

cmp相当于sub，只设置条件码不改变目的寄存器，不需要考虑条件码的实际应用场景，只做位级的减法运算

区分无符号数和有符号数在jxx实现

jg	*Label*	jnle	~(SF ^ OF) & ~ZF	Greater (signed >)
jge	*Label*	jnl	~(SF ^ OF)	Greater or equal (signed >=)
jl	*Label*	jnge	SF ^ OF	Less (signed <)
jle	*Label*	jng	(SF ^ OF) | ZF	Less or equal (signed <=)
ja	*Label*	jnbe	~CF & ~ZF	Above (unsigned >)
jae	*Label*	jnb	~CF	Above or equal (unsigned >=)
jb	*Label*	jnae	CF	Below (unsigned <)
jbe	*Label*	jna	CF | ZF	Below or equal (unsigned <=)

Q12

4. 对于如下的C语言中的条件转移指令，它所对应的汇编代码中至少包含几条条件转移指令：`if (a > 0 && a != 1 || a < 0 && a != -1) b=a;`

A. 2条　　B. 3条　　C. 4条　　D. 5条

B

ps: &&和||的优先级不同，先算&&

伪汇编如下

1: if a==1 jump to 5

2: if a==-1 jump to 5

3: if a==0 jump to 5

4: b=a

5:

B

Q13

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

B

Jump的目标地址等于Jump下一条指令的地址（当前%rip地址寄存器的值，具体会在下一章流水线中学习）加上反汇编的第二个数，

Jump下一条指令的地址为pop %rbp的地址，即400022

F0=11110000(2进制)=-16（十进制）=-10（十六进制）

400022-10=400012

B

Q14

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

C

'0' = $48 = 0x30 (ASCII)

有个sub $48，所以x在'0'~'9'考虑

L2是default块，去掉jumpL2还剩'0','3','4','5','6','8'

C

Q15

5. 在下列关于条件传送的说法中，正确的是：

A. 条件传送可以用来传送字节、字、双字、和 4 字的数据

B. C 语言中的“?:”条件表达式都可以编译成条件传送

C. 使用条件传送总可以提高代码的执行效率

D. 条件传送指令不需要用后缀（例如 b, w, l, q）来表明操作数的长度

D

长度不能是Byte，只能是w,l,q。

可能存在副作用或者空指针，会导致程序错误。

当待选的两个值自身求值需要很长时间时，条件传送更劣。

cmov不需要加长度，可以自行适配。

[Extra Content] MOV要求明确指定长度，CMOV可以通过寄存器名称推断出操作数长度
https://en.wikipedia.org/wiki/X86_instruction_listings
MOV 来自最早的8086/8088 指令集（1976/1978/1979）
CMOV 来自Pentium 5/6代 指令集（支持SSE前）（1993/1995）

D

Q16

4. 以下关于 x86-64 指令的描述，说法正确的有几项？

   a) 有符号除法指令 idivq S 将%rdx（高 64 位）和%rax（低 64 位）中的 128 位数作为被除数，将操作数 S 的值作为除数，做有符号除法运算；指令将商存在%rdx 寄存器中，将余数存在%rax 寄存器中。

   b) 我们可以使用指令 jmp %rax 进行间接跳转，跳转的目标地址由寄存器%rax 的值给出。

   c) 算术右移指令 shr 的移位量既可以是一个立即数，也可以存放在单字节寄存器%cl 中。

   d) leaq 指令不会改变任何条件码。

   A. 1
   B. 2
   C. 3
   D. 4

a) ×
idivq中，
商在%rax 余数在%rdx

b) ×
间接跳转：jmp *Operand
此处应为jmp *%rax

c) ×
a = arithmetic
sar = 算术右移
shr = 逻辑右移

d) √

A

A