

ICS Midterm Review



Yuxuan Kuang
School of EECS, PKU

2022-10-26

Basic

1. Bit-level operations
2. Integers
3. Floating-point numbers

Enhanced

1. Big Endian and Little Endian, **string**
2. Casting: change the way of interpreting, **implicit**
3. Overflow and roundings
4. Special floating point numbers: **norm, denorm, inf, nan**



3. 运行下面的代码，输出结果是（其中 `float` 类型表示 IEEE-754 规定的浮点数，包括 1 位符号、8 位阶码和 23 位尾数）：

```
for(float f = 1;; f = f + 1)
{
    if(f + 1 - f != (float)1)
    {
        printf("%.0f\n", f);
        break;
    }
}
```

- A. 8388608 ($=2^{23}$)
- B. 16777216 ($=2^{24}$)
- C. 2147483647 ($=2^{31}-1$)
- D. 程序为死循环，没有输出



3. 运行下面的代码，输出结果是（其中 float 类型表示 IEEE-754 规定的浮点数，包括 1 位符号、8 位阶码和 23 位尾数）：

```
for(float f = 1;; f = f + 1)
{
    if(f + 1 - f != (float)1)
    {
        printf("%.0f\n", f);
        break;
    }
}
```

- A. 8388608 ($=2^{23}$)
B. 16777216 ($=2^{24}$)
C. 2147483647 ($=2^{31}-1$)
D. 程序为死循环，没有输出

B



Basic

1. Program Encodings, registers
2. Control
3. Procedures
4. Data

Enhanced

1. Alignment
2. Arrays and Pointers, **complex and nested**
3. Condition Codes: instructions*, combinations
4. Misc: **sizeof***, **jump*** & **switch**, **RISC & CISC**, **gdb**



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



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

A



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 `jb` 时，那么发生跳转等价于以下哪一个表达式的结果为 1？
- A. $\sim(SF \wedge OF) \ \& \ \sim ZF$
 - B. $\sim(SF \wedge OF)$
 - C. $SF \wedge OF$
 - D. $(SF \wedge OF) \mid ZF$



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 `jb` 时，那么发生跳转等价于以下哪一个表达式的结果为 1？
- A. $\sim(SF \wedge OF) \ \& \ \sim ZF$
 - B. $\sim(SF \wedge OF)$
 - C. $SF \wedge OF$
 - D. $(SF \wedge OF) \mid ZF$

A



Basic

1. Instruction Set Architecture
2. Sequential Logic
3. Pipeline

Enhanced: new instructions

1. Stages
2. Pipeline correction, HCL
3. Hazards



11. 如下图所示，①~④为四个组合逻辑单元，对应的延迟已在图上标出，REG0 为一寄存器，延迟为 20ps。通过插入**额外的 2 个**流水线寄存器 REG1、REG2（延迟均为 20ps），可以对其进行流水化改造。改造后的流水线的吞吐率最大为 _____ GIPS。



- A. 7.69 B. 8.33 C. 10.00 D. 11.11



11. 如下图所示，①~④为四个组合逻辑单元，对应的延迟已在图上标出，REG0 为一寄存器，延迟为 20ps。通过插入**额外的 2 个**流水线寄存器 REG1、REG2（延迟均为 20ps），可以对其进行流水化改造。改造后的流水线的吞吐率最大为 _____ GIPS。



- A. 7.69 B. 8.33 C. 10.00 D. 11.11

C



New instruction: `cretXX`

9	fun
---	-----

 (32 bits)

Stage	cretXX Offset
Fetch	
Decode	
Execute	
Memory	
Write back	
PC update	



New instruction: **cretXX**

9	fun
---	-----

 (32 bits)

Stage	cretXX Offset
Fetch	<u>$\text{icode:ifun} \leftarrow M_1[\text{PC}]$</u> <u>$\text{valP} \leftarrow \text{PC}+1$</u>
Decode	<u>$\text{valB} \leftarrow R[\%esp]$</u> <u>$\text{valA} \leftarrow R[\%esp]$</u>
Execute	<u>$\text{valE} \leftarrow \text{valB} + 4$</u> <u>$\text{Cnd} \leftarrow \text{Cond}(\text{CC}, \text{ifun})$</u>
Memory	<u>$\text{valM} \leftarrow M_4[\text{valA}]$</u>
Write back	<u>$\text{if } (\text{Cnd}) R[\%esp] \leftarrow \text{valE}$</u>
PC update	<u>$\text{PC} \leftarrow \text{Cnd} ? \text{valM} : \text{valP}$</u>



Basic

1. Performance Metrics and Analysis
2. Loop Unrolling & Parallelism

Enhanced

1. Modern Processors
2. Memory Performance



11. 假设已有声明 `int i, int sum, int *p, int *q, int *r, const int n = 100, float a[n], float b[n], float c[n], int foo(int), void bar()`, 以下哪项程序优化编译器总是可以进行?

A	<pre>for(i = 0; i < n; ++i) { a[i] += b[i]; a[i] += c[i]; }</pre>	<pre>float tmp; for(i = 0; i < n; ++i) { tmp = b[i] + c[i]; a[i] += tmp; }</pre>
B	<pre>*p += *q; *p += *r;</pre>	<pre>int tmp; tmp = *q + *r; *p += tmp;</pre>
C	<pre>for(i = 0; i < n; ++i) sum += i * 4;</pre>	<pre>int N = n * 4; for(i = 0; i < N; i += 4) sum += i;</pre>
D	<pre>for(i = 0; i < foo(n); ++i) bar();</pre>	<pre>int tmp = foo(n); for(i = 0; i < tmp; ++i) bar();</pre>



11. 假设已有声明 `int i, int sum, int *p, int *q, int *r, const int n = 100, float a[n], float b[n], float c[n], int foo(int), void bar()`, 以下哪项程序优化编译器总是可以进行?

A	<pre>for(i = 0; i < n; ++i) { a[i] += b[i]; a[i] += c[i]; }</pre>	<pre>float tmp; for(i = 0; i < n; ++i) { tmp = b[i] + c[i]; a[i] += tmp; }</pre>
B	<pre>*p += *q; *p += *r;</pre>	<pre>int tmp; tmp = *q + *r; *p += tmp;</pre>
C	<pre>for(i = 0; i < n; ++i) sum += i * 4;</pre>	<pre>int N = n * 4; for(i = 0; i < N; i += 4) sum += i;</pre>
D	<pre>for(i = 0; i < foo(n); ++i) bar();</pre>	<pre>int tmp = foo(n); for(i = 0; i < tmp; ++i) bar();</pre>

C



Basic

1. Memory Hierarchy
2. Caches

Enhanced

1. Cache Organization: simulations and optimizations
2. Locality
3. Substitution Policies



14. 以下关于存储的描述中，正确的是（ ）

- A) 由于基于 SRAM 的内存性能与 CPU 的性能有很大差距，因此现代计算机使用更快的基于 DRAM 的高速缓存，试图弥补 CPU 和内存间性能的差距。
- B) SSD 相对于旋转磁盘而言具有更好的读性能，但是 SSD 写的速度通常比读的速度慢得多，而且 SSD 比旋转磁盘单位容量的价格更贵，此外 SSD 底层基于 EEPROM 的闪存会磨损。
- C) 一个有 2 个盘片、10000 个柱面、每条磁道平均有 400 个扇区，每个扇区有 512 个字节的双面磁盘的容量为 8GB。
- D) 访问一个磁盘扇区的平均时间主要取决于寻道时间和旋转延迟，因此一个旋转速率为 6000RPM、平均寻道时间为 9ms 的磁盘的平均访问时间大约为 19ms。



14. 以下关于存储的描述中，正确的是（ ）

- A) 由于基于 SRAM 的内存性能与 CPU 的性能有很大差距，因此现代计算机使用更快的基于 DRAM 的高速缓存，试图弥补 CPU 和内存间性能的差距。
- B) SSD 相对于旋转磁盘而言具有更好的读性能，但是 SSD 写的速度通常比读的速度慢得多，而且 SSD 比旋转磁盘单位容量的价格更贵，此外 SSD 底层基于 EEPROM 的闪存会磨损。
- C) 一个有 2 个盘片、10000 个柱面、每条磁道平均有 400 个扇区，每个扇区有 512 个字节的双面磁盘的容量为 8GB。
- D) 访问一个磁盘扇区的平均时间主要取决于寻道时间和旋转延迟，因此一个旋转速率为 6000RPM、平均寻道时间为 9ms 的磁盘的平均访问时间大约为 19ms。

B



13. 设一种全相联缓存共包含 4 个缓存块,如果循环地顺序访问 5 个不同的缓存块,下列哪种替换算法会产生最多的命中?
- A) 最近最少使用替换策略 LRU
 - B) 先入先出替换策略 FIFO
 - C) 随机替换策略 Random
 - D) 后入先出替换策略 LIFO



13. 设一种全相联缓存共包含 4 个缓存块,如果循环地顺序访问 5 个不同的缓存块,下列哪种替换算法会产生最多的命中?
- A) 最近最少使用替换策略 LRU
 - B) 先入先出替换策略 FIFO
 - C) 随机替换策略 Random
 - D) 后入先出替换策略 LIFO

CD



Thank you for your attention!

