

哈尔滨工业大学（深圳）  
2025 年春《计算机系统》期末考试试题（A）  
题目来源：群友 排版：Chi. Ya.

一、选择题（共 20 分，每题 2 分）

1. 虚拟机是对 \_\_\_\_\_ 的抽象
  - A. I/O 设备
  - B. 处理器硬件
  - C. 操作系统
  - D. 整个计算机
  
2. 下列说法正确的是 \_\_\_\_\_
  - I. 浮点数不是实数
  - II. 浮点加法有结合律，乘法也有结合律
  - III. 浮点加法无结合律，乘法有结合律
  - IV. 浮点加法有结合律，乘法无结合律
  - V. 浮点数运算无分配率  
  - A. I,II,V
  - B. I,III,V
  - C. I,IV,V
  - D. I,V
  
3. 以下哪个条件可以判断补码加法  $s = x + y$  发生了负溢出？ \_\_\_\_\_
  - A.  $x > 0, y > 0, s \leq 0$
  - B.  $x > 0, y < 0, s \geq 0$
  - C.  $x < 0, y > 0, s \leq 0$
  - D.  $x < 0, y < 0, s \geq 0$
  
4. 

```
short t = -8196;  
int y = t;
```

  
y 的十六进制表示为 \_\_\_\_\_  
  - A. 0x00009FFC
  - B. 0x0000DFFC
  - C. 0xFFFF9FFC
  - D. 0xFFFFDFFC

5. 已知依赖 `fun.o -> libx.a, liby.a; libx.a -> libz.a; libx.a` 与 `liby.a` 相互独立, `libz.a` 与 `liby.a` 相互独立, 则下面给出的编译命令中, 会发生错误的是: \_\_\_\_\_
- A. `gcc -static -Og fun.o libx.a liby.a libz.a`
  - B. `gcc -static -Og fun.o liby.a libz.a libx.a`
  - C. `gcc -static -Og fun.o libx.a libz.a liby.a`
  - D. `gcc -static -Og fun.o liby.a libx.a libz.a`
6. 由于 C 语言提供指针运算, 因此 \_\_\_\_\_ 现象会导致潜在的性能下降和正确性下降
- A. 非法指针
  - B. 函数调用
  - C. 内存别名
  - D. 缺页
7. 以下 \_\_\_\_\_ 属于异步异常
- A. 计时器中断
  - B. 缺页
  - C. 系统调用
  - D. 非法指令
8. 关于 `fork()` 的调用, 错误的是 \_\_\_\_\_
- A. 调用一次, 返回两次
  - B. 父子进程间运行顺序不可预测
  - C. 对子进程返回 0, 对父进程返回子进程的进程号 `pid`
  - D. 子进程和父进程执行完全相同的代码直到进程结束
9. 关于虚拟地址空间的叙述, 不正确的是 \_\_\_\_\_
- A. DRAM 空间可以理解为虚拟地址的缓存
  - B. 每个程序都有独立的虚拟地址空间
  - C. 虚拟地址直接映射到物理地址的连续区域
  - D. 虚拟内存增强了进程的局部性
10. 关于动态库, 以下说法不正确的是 \_\_\_\_\_
- A. 可在加载时链接, 即当可执行文件首次加载和运行时进行动态链接。
  - B. 当动态库更新但接口不变时, 使用它的程序无需重新编译
  - C. 动态库可在编译期与装载期链接, 不能在运行时链接
  - D. 即便有多个正在运行的程序使用同一动态库, 系统也仅在内存中载入一份动态库。

二、填空题（共 10 分，每空 2 分）

1. 写出将 `int` 变量 `x` 的最高有效字节清零的 C 表达式: \_\_\_\_\_ (限一个表达式)。
2. 64 位系统中 `struct node{ int a; char c1; char c2; short x; float *p;};`,  
则 `sizeof(node)` 的值为 \_\_\_\_\_。
3. Unix 将 I/O 设备抽象为: \_\_\_\_\_。
4. 要将程序的标准输出 `stdout` (文件描述符为 1) 重定向到 `file.txt` (设其文件描述符为 4),  
内核调用的函数是 \_\_\_\_\_ (包括函数名和参数)。
5. 虚拟页面的状态有 \_\_\_\_\_、已缓存、未缓存。

三、简答题（共 15 分）

1. 简述缓冲区溢出的原理及防范方法。(5 分)

2. 由以下 C 语言代码 `hello.c` 生成可执行文件需要经历哪几个步骤？描述每个步骤发生了什么变化。（5 分）

```
#include <stdio.h>
int main(void) {
    printf("hello world\n");
}
```

3. 列举三种常见的程序优化方法，并任选一种结合代码示例分析。（5 分）

四、计算题（共 10 分）

给出 C 语言变量 `float a = -2.1`

1. 给出 a 的十六进制表示
2. 再给定 `float b = 1.05`; 两数的二进制表示有多少个位不同?

## 五、程序分析题（共 30 分）

### 1. 分析以下 Y86-64 命令

```
0x006  irmovq $10, %rsp
____  call Label
____  addq %rdx, %rax
____  ret
```

(1) 在表中填写空缺的指令地址。(3 分)

(2) 第一条指令和第二条指令之间是否有冒险。如有，给出具体的处理冒险的解决方案。(4 分)

(3) 补充 `ret` 的 Y86-64 微指令。(8 分)

指令	<code>ret</code>
取指	<code>icode : ifun ← M<sub>1</sub>[PC]</code>  $valP ← PC + 1$
译码	$valA ← R[%rsp]$ $valB ← R[%rsp]$
执行	_____
访存	_____
写回	_____
更新 PC	_____

2. 分析以下 x86-64 指令 (4 分)

```
.varx
    int 123, -2345    # 123 的地址是 varx, -2345 在 varx+4

    movq $varx, %rax
    movl 4(%rax), %edx
    movl $-1, %rax
```

(1) 执行 `movl 4(%rax), %edx` 后 `%edx` = \_\_\_\_\_ (long long int 十进制表示)

(2) 执行 `movl $-1, %rax` 后 `%rax` = \_\_\_\_\_ (long long int 十进制表示)

3. 分析以下 x86-64 机器上的代码 (6 分)

```
struct rec {
    int a[4];
    int i;
    struct rec* next;
}

void fun(struct rec* r, int val) {
    while (r) {
        int i = r->i;
        a[i] = val;
        r = r->next;
    }
}
```

下面是实现上述循环的汇编代码，代码中存在错误，在错误的指令右边写出正确的汇编指令  
(注意：无需添加寄存器)。

```
jmp L12
.L11:
    movslq 16(%esi), %rax
    movl %esi, (%edi, %eax, 4)
    movq 20(%edi), %edi
.L12:
    testq %edi, %edi
    je L11
```

4. (5 分)

```
int main() {
    int status, count = 0;
    int pid = fork();
    if (pid == 0) {
        printf("%d\n", --count);
        exit(0);
    }
    else {
        printf("%d\n", ++count);
        wait(&status);
        printf("%d\n", ++count);
    }
    printf("%d\n", ++count);
}
```

(1) 共有几种输出可能? (1 分)

(2) 写出所有可能的输出序列。 (4 分)

## 六、综合分析题（共 15 分，每空 1 分）

某计算机系统的情况如下：

1. 内存是按字节寻址，字长为 2 字节（非 4 字节），采用小端模式存储；
2. 虚拟地址长度：24 位；
3. 使用一级页表，页面大小是 2048 字节；
4. 物理地址的位数是 19 位；
5. TLB（翻译后备缓冲器）是 4 路组相连，共 16 个条目；
6. L1 d-Cache 是物理寻址、直接映射（每组一行）、行大小 8 字节，共 8 个组；
7. TLB、Cache 的当前数值如表 1、2 所示。

表 1：TLB 数值表

组	标记位	PPN	有效位									
0	0CD	09	1	AA1	00	1	3E0	62	1	C4C	48	1
1	312	45	0	010	75	1	987	3A	1	D39	3F	0
2	038	E3	0	0A7	13	0	18B	52	1	49B	11	0
3	6C0	42	0	075	50	0	013	39	1	0F2	0D	0

表 2：L1 d-Cache 的数值

索引	标记位	有效位	块 0	块 1	块 2	块 3	块 4	块 5	块 6	块 7
0	35E	1	42	A0	75	50	42	00	05	50
1	27B	1	08	E3	00	A7	13	00	88	52
2	C54	1	3F	75	AB	11	25	78	9A	00
4	A32	1	97	3A	91	D3	3F	12	86	22
5	C30	1	30	62	15	4C	48	A1	12	5C
6	B26	1	01	25	3E	62	1F	C4	85	12
7	01A	1	98	3A	12	D3	3F	3C	4D	5E

- (1) VPN 的位数是 \_\_\_\_\_ 位，页表项 PTE 的总数量是 \_\_\_\_\_ 个。
- (2) TLB 中，组索引是 \_\_\_\_\_ 位，标记 Tag 的位数是 \_\_\_\_\_ 位。
- (3) 在 L1 d-Cache 中，块偏移的位数是 \_\_\_\_\_ 位，组索引的位数是 \_\_\_\_\_ 位，标记的位数是 \_\_\_\_\_ 位。
- (4) CPU 从虚拟地址 0x7C0515 读取一个字的数值，将虚拟地址翻译成物理地址，并获取数值的过程中，写出下表中各参数对应的数值（8 分，每空 1 分）。

参数	数值
虚拟地址	0x7C0515
虚拟页号 VPN	_____
TLB 索引	_____
TLB 标签 Tag	_____
TLB 命中? (Y/N)	_____
物理页号 PPN	_____
物理地址	_____
Cache 命中? (Y/N)	_____
读取的数值	_____