

考试中心填写：

—年 —月 —日
考试用

湖南大学课程考试试卷

课程名称：计算机组成与结构 B(2017 春)； 试卷编号：A； 考试时间：120 分钟

题号	一	二	三	四	五	六	七	八	九	十	总分
应得分	6	6	38	10	15	25					100
实得分											
评卷人											评分：

一. (6 分, 每空 0.5 分) 下表中 %r1, %r2 为两个四位的寄存器, 请仿照第一行填写在执行指令 `cmp %r1, %r2` 后, ZF, SF, OF, CF 各标志位的值, 并判断如果紧接着执行 JA 指令或 JG 指令时是否会跳转。

%r1	%r2	ZF	SF	OF	CF	JA 跳转?	JG 跳转?
0x0001	0x0001	1	0	0	0	否	否
0x0111	0x1100						
0x1110	0x0100						

二. (6 分, 第二列每空 0.5 分, 第三列每空 1 分) 我们在一个 int 类型为 32 位补码表示的机器上运行程序, float 类型的值使用 32 位 IEEE 格式, 而 double 类型使用 64 位 IEEE 格式, x, y, z 为随机产生的 32 位 int 型整数, 下面代码把它们转换成 double 类型的值:

```
double dx=(double) x;
```

```
double dy=(double) y;
```

```
double dz=(double) z;
```

请填写下表:

表达式	是否总为 1	理由
<code>(double) (float) x==dx</code>	否	32 位 float 型时无法准确表示所有 32 位整数, 而 double 可以
<code>dx+dy == (double) (x+y)</code>		
<code>dx+dy+dz == dz+dy+dx</code>		
<code>dx*dy*dz == dz*dy*dx</code>		
<code>dx/dx == dy /dy</code>		

三. (40 分) 以下有三段完整或者不完整的 C 程序段及相应的汇编代码 (在 32 位环境下), 请填空。

(1) (8 分, 每空 4 分)

C 代码:

```
#include "stdio.h"
```

```
#define H ? //H, J 为 define 定义的常数
```

```
#define J ?
```

```
int A[H][J];
```

```
int B[J][H];
```

```
int C[H][H];
```

```
void f(int x,int y,int z) {
```

```
    int i=0;
```

```
    for(i=0;i<z;i++)
```

```
        C[x][y]+=A[x][i]*B[i][y];
```

```
}
```

```
int main()
```

```
{
```

```
    return 0;
```

```
}
```

在 32 位环境下使用 gcc -S 编译后查看函数 f 对应的代码如下所示:

f:

```
    pushl    %ebp
```

```
    movl    %esp, %ebp
```

```
    pushl    %edi
```

```
    pushl    %esi
```

```
    pushl    %ebx
```

```
    subl    $16, %esp
```

```
    movl    $0, -16(%ebp)
```

```
    movl    $0, -16(%ebp)
```

```
    jmp     .L2
```

```
.L3:
```

```
    movl    8(%ebp), %ebx
```

```

movl    12(%ebp), %ecx
movl    8(%ebp), %edx
movl    12(%ebp), %eax
sall    $2, %edx
leal    (%edx,%eax), %eax
movl    C(%eax,4), %esi
movl    8(%ebp), %edx
movl    -16(%ebp), %edi
movl    %edx, %eax
sall    $4, %eax
addl    %edx, %eax
addl    %edi, %eax
movl    A(%eax,4), %edx
movl    -16(%ebp), %edi
movl    12(%ebp), %eax
sall    $2, %edi
leal    (%edi,%eax), %eax
movl    B(%eax,4), %eax
imull   %edx, %eax
leal    (%esi,%eax), %edx
leal    0(%ebx,4), %eax
addl    %ecx, %eax
movl    %edx, C(%eax,4)
addl    $1, -16(%ebp)

```

.L2:

```

movl    -16(%ebp), %eax
cmpl    16(%ebp), %eax
jl      .L3
addl    $16, %esp
popl    %ebx
popl    %esi
popl    %edi
popl    %ebp
ret

```

由此可以推出

H = _____

J = _____

(2) (10 分) 如下为某个对数组排序的 C 语言函数及 32 位环境下对应的汇编代码，请根据汇编代码补充完整 C 语言函数。

C 代码:

```
void selection_sort(int a[], int n) {  
    int i, t, imax = 0;  
    if (____A____) return;  
    for(i = 1; i < n; ++i) {  
        if(____B____)  
            ____C____;  
    }  
    If (____D____) {  
        t = a[n - 1];  
        a[n - 1] = a[imax];  
        a[imax] = t;  
    }  
    ____E____;  
}
```

此函数所对应的汇编代码:

selection_sort:

```
    pushl    %ebp  
    movl     %esp, %ebp  
    subl     $40, %esp  
    movl     $0, -20(%ebp)  
    cmpl     $0, 12(%ebp)  
    jle      .L9
```

.L2:

```
    movl     $1, -12(%ebp)  
    jmp      .L4
```

.L6:

```
    movl     -20(%ebp), %eax  
    sall     $2, %eax
```

```

    addl    8(%ebp), %eax
    movl    (%eax), %edx
    movl    -12(%ebp), %eax
    sall    $2, %eax
    addl    8(%ebp), %eax
    movl    (%eax), %eax
    cmpl    %eax, %edx
    jge     .L5
    movl    -12(%ebp), %eax
    movl    %eax, -20(%ebp)
.L5:
    addl    $1, -12(%ebp)
.L4:
    movl    -12(%ebp), %eax
    cmpl    12(%ebp), %eax
    jl      .L6
    movl    12(%ebp), %eax
    subl    $1, %eax
    cmpl    -20(%ebp), %eax
    je      .L7
    movl    12(%ebp), %eax
    subl    $1, %eax
    sall    $2, %eax
    addl    8(%ebp), %eax
    movl    (%eax), %eax
    movl    %eax, -16(%ebp)
    movl    12(%ebp), %eax
    subl    $1, %eax
    sall    $2, %eax
    addl    8(%ebp), %eax
    movl    -20(%ebp), %edx
    sall    $2, %edx
    addl    8(%ebp), %edx
    movl    (%edx), %edx

```

```

        movl    %edx, (%eax)
        movl    -20(%ebp), %eax
        sall    $2, %eax
        addl    8(%ebp), %eax
        movl    -16(%ebp), %edx
        movl    %edx, (%eax)
.L7:
        movl    12(%ebp), %eax
        subl    $1, %eax
        movl    %eax, 4(%esp)
        movl    8(%ebp), %eax
        movl    %eax, (%esp)
        call    selection_sort
        jmp     .L8
.L9:
        nop
.L8:
        leave
        ret

```

其中 A, B, C, D, E 处所填代码应为: (每空 2 分)

A: _____

B: _____

C: _____

D: _____

E: _____

(3) (20 分) 请仔细阅读如下 C 语言代码及 32 位环境下对应的汇编代码。

C 代码:

```

int fact(int n)
{
    if (n == 1)

```

```

        return n;

    else

        return n * fact(n-1);

}

```

汇编代码如下：

```

080483a4    <fact>:
80483a4:    55                push %ebp
80483a5:    89  e5            mov %esp,%ebp
80483a7:    53                push %ebx
80483a8:    83  ec  04        sub $0x4,%esp
80483ab:    8b  5d  08        mov 0x8(%ebp),%ebx
80483ae:    83  fb  01        cmp $0x1,%ebx
80483b1:    74  0e            je  80483c1 <fact+0x1d>
80483b3:    8d  43  ff        lea 0xffffffff(%ebx),%eax
80483b6:    89  04  24        mov %eax,(%esp)
80483b9:    e8  e6  ff  ff  ff call 80483a4 <fact>
80483be:    0f  af  d8        imul %eax,%ebx
80483c1:    89  d8            mov %ebx,%eax
80483c3:    83  c4  04        add $0x4,%esp
80483c6:    5b                pop %ebx
80483c7:    5d                pop %ebp
80483c8:    c3                ret

```

某函数调用了 fact (5) 后，由程序代码可知，代码将调用 fact (4), fact (3)、fact (2) 直至 fact (1)，请填写正好开始调用 fact (3) 时的栈帧中内容（即下图中的字母处内容）。并给出此时的当前 %ebp 寄存器与 %esp 寄存器中的值。

假设在调用 fact (5) 时，其调用函数的 %ebp 值为 0xffffd848，返回地址为：0x080483e6。

在作答时，请按如下格式填写内容（斜体值内容可以改变）：

返回地址值：0x080483e6

旧 %ebp 值：0xffffd848

n 当前值 5

寄存器 `%eax` 的当前值

	+-----+	
0xffffd830	n 当前值: 5	调用 <code>fact(5)</code> 开始
	+-----+	
0xffffd82c	A	
	+-----+	
0xffffd828	B	
	+-----+	
0xffffd824	C	
	+-----+	
0xffffd820	D	
	+-----+	
0xffffd81c	E	
	+-----+	
0xffffd818	F	
	+-----+	
0xffffd814	G	
	+-----+	
0xffffd810	H	
	+-----+	

(每空 2 分)

A: _____ E: _____

B: _____ F: _____

C: _____ G: _____

D: _____ H: _____

当前 `%ebp` 值: 0x 当前 `%esp` 值: 0x

四 (10 分)、给定如下基本数据结构, 请分析代码并回答问题。


```

1  /* Create abstract data type for vector */
2  typedef struct {
3      long int len;
4      data_t *data;
5  } vec_rec, *vec_ptr;

```



代码如下：

```

1  void combine(vec_ptr v, data_t *dest)
2  {
3      long int i;
4      long int length=vec_length(v);    //获取向量长度;
5      long int limit=length-1;
6      data_t *data=get_vec_start(v);    //获取向量元素;
7      data_t acc=IDENTj;    //加法为 0，乘法为 1;
8      for (i=0;i<limit; i+=2){
9          acc=acc OP (data[i] OP data[i+1]); //OP 为加法或者乘法操作符;
10         }
11     for (;i<length;i++){
12         acc=acc OP data[i];
13     }
14     *dest=acc;
15 }

```

请回答：

(1) 代码第 8 行、第 9 行展示了哪两种常规优化技术？（4 分）

(2) 若第 9 行代码替换为 $acc = (acc \text{ OP } data[i]) \text{ OP } data[i+1]$ 时，对于操作数分别为整数和浮点数，编译后的程序展现出来的性能与替换前有何不同。试分析之（6 分）。

五. (15 分) 有如下 cache 系统：存储器是按字节寻址，并且按 1 字节进行访问；主存容量 8K byte；高速缓存是 2 路组相联的、块大小为 4 字节、8 个组；高速缓存中数据内容如下：

组索引	标记	有效位	字节 0	字节 1	字节 2	字节 3
0	00	1	40	41	42	43
	83	1	FE	97	CC	D0
1	00	1	44	45	46	47
	83	0	-	-	-	-
2	00	1	48	49	4A	4B
	40	0	-	-	-	-
3	FF	1	9A	C0	03	FF
	00	0	-	-	-	-
4	00	1	48	49	4A	4B
	40	0	-	-	-	-
5	FF	1	9A	C0	03	FF
	00	0	-	-	-	-
6	91	1	48	49	4A	4B
	40	0	-	-	-	-
7	FF	1	9A	C0	03	FF
	00	0	-	-	-	-

(1) cache 的容量为多少？（2 分），并请给出主存地址各个字段的组成格式（3 分）；

(2) 列出所有会在组 4 中命中的地址，必须有具体说明分析过程（10 分）。

六、阅读如下代码机器参考输出，回答如下问题：

```
1  #include "csapp.h"
2
3  void handler1(int sig)
4  {
5      pid_t pid;
6
7      if ((pid = waitpid(-1, NULL, 0)) < 0)
8          unix_error("waitpid error");
9      printf("Handler reaped child %d\n", (int)pid);
10     Sleep(2);
11     return;
12 }
13
14 int main()
15 {
16     int i, n;
17     char buf[MAXBUF];
18
19     if (signal(SIGCHLD, handler1) == SIG_ERR)
20         unix_error("signal error");
21
22     /* Parent creates children */
23     for (i = 0; i < 3; i++) {
24         if (Fork() == 0) {
25             printf("Hello from child %d\n", (int)getpid());
26             Sleep(1);
27             exit(0);
28         }
29     }
30
31     /* Parent waits for terminal input and then processes it */
32     if ((n = read(STDIN_FILENO, buf, sizeof(buf))) < 0)
33         unix_error("read");
34
35     printf("Parent processing input\n");
36     while (1)
37         ;
38
39     exit(0);
40 }
```

左侧代码的参考输出结果如下示：

```
linux> ./signal1
Hello from child 10320
Hello from child 10321
Hello from child 10322
Handler reaped child 10320
Handler reaped child 10322
<cr>
Parent processing input

<ctrl-z>
Suspended
linux> ps
PID TTY STAT TIME COMMAND
...
10319 p5 T   0:03 signal1
10321 p5 Z   0:00 signal1 <defunct>
10323 p5 R   0:00 ps
```

(1) 简析这段代码中 fork 函数的作用和特点；（5 分）

(2) 如上代码是一段有问题的代码，请分析哪里出了问题？为什么会出现这类问题？从中获得的教训是什么？（10 分）

(3) 从虚存角度，阐释 `fork` 函数如何创建一个新进程。(10 分)