

计算机系统试题

主管
领导
审核
签字

题号	一	二	三	四	五	总分
得分						
阅卷人						

考生须知：本次考试为闭卷考试，考试时间为 120 分钟，满分 100 分。

一、单项选择题（每小题 1 分，共 10 分）

- 在 x86-64 系统中，调用函数 `int exmp(long x, long y, long z)` 时，保存参数 `x` 的寄存器是（ **D** ）
A. `%rcx` B. `%rsi` C. `%rdx` D. `%rdi`
- C 语言程序中的整数常量、整数常量表达式是在（ **B** ）阶段变成 2 进制补码的
A. 预处理 B. 编译 C. 链接 D. 执行
- 以下有关编程语言的叙述中，正确的是（ **D** ）
A. 计算机能直接执行高级语言程序和汇编语言程序
B. 机器语言可以通过汇编过程变成汇编语言
C. 汇编语言比高级语言有更好的可读性
D. 汇编语言和机器语言都与计算机系统结构相关
- C 语言中不同类型的数值进行强制类型转换时，下列说法正确的是（ **C** ）
A. 从 `int` 转换成 `float` 时，数值可能会溢出
B. 从 `int` 转换成 `double` 后，数值虽然不会溢出，但有可能是不精确的
C. 从 `double` 转换成 `float` 时，数值可能会溢出
D. 从 `double` 转换成 `int` 时，数值不可能溢出
- 下列哪项不是 Y86-64 流水线 CPU 中的冒险的种类（ **A** ）
A. 流量冒险 B. 数据冒险 C. 控制冒险 D. 加载使用冒险
- C 程序执行到整数或浮点变量除以 0 可能发生（ **D** ）
A. 显示除法溢出错误直接退出 B. 程序不提示任何错误
C. 可由用户程序确定处理办法 D. 以上都可能
- 关于局部变量，正确的叙述是（ **D** ）
A. 普通（`auto`）局部变量也是一种编程操作的数据，存放在数据段。
B. 非静态局部变量在链接时是本地符号。
C. 静态局部变量是全局符号。
D. 编译器可将 `%rsp` 减去一个数为局部变量分配空间。
- 虚拟内存发生缺页的时候，正确的叙述是（ **C** ）
A. 当发生虚拟内存缺页的时候，程序会直接退出。
B. 缺页产生的中断请求通常由 CPU 产生。
C. 当处理程序处理完缺页错误之后，会重新执行引发缺页的命令。
D. 当一个程序先后重复执行两次的时候，会在相同的指令位置产生缺页错误。
- 虚拟内存系统中的实际物理地址和虚拟地址之间的关系是（ **C** ）
A. 1 对 1 B. 多对 1 C. 1 对多 D. 多对多

授课教师

姓名

学号

院系

封

二、填空题（每空 2 分，共 10 分）

10. C 语言程序定义了结构体 `struct noname{int *a; char b;short c;}`；若该程序编译成 64 位可执行程序，则 `sizeof(noname)` 的值是_____。
11. -1024 采用 IEEE 754 单精度浮点数格式表示的结果是_____（16 进制形式）。
12. 下面是 Y86-64 中的一段汇编程序，请指出 `jne t` 指令之后执行的那条指令的地址是_____（16 进制表示）。
- ```
0x000: xorq %rax,%rax
0x002: jne t
...

0x019: t: irmovq $3, %rdx
0x023: irmovq $4, %rcx
0x02d: irmovq $5, %rdx
```
13. 进程创建函数 `fork` 执行成功后返回\_\_\_\_\_次。
14. 对于一个磁盘，其平均旋转速率是 3000 RPM，平均寻道时间是 10ms，单个磁道上平均扇区数量是 1600，则这个磁盘的平均访问时间是\_\_\_\_\_。

## 三、系统分析题（每题 5 分，共 20 分）

15. 阅读 `sum` 函数反汇编结果，解释 1-5 每行指令的功能和作用。（10 分）

4004e7 <sum>:

`push %rbp` #①

`mov %rsp,%rbp` #②

`mov %rdi,-0x4(%rbp)` #③

`jmp 400512 <sum+0x2b>`

4004f4:

`mov -0x4(%rbp),%eax`

`cltq`

`mov 0x601030(,%rax,4),%edx`

授课教师

姓名

学号

院系

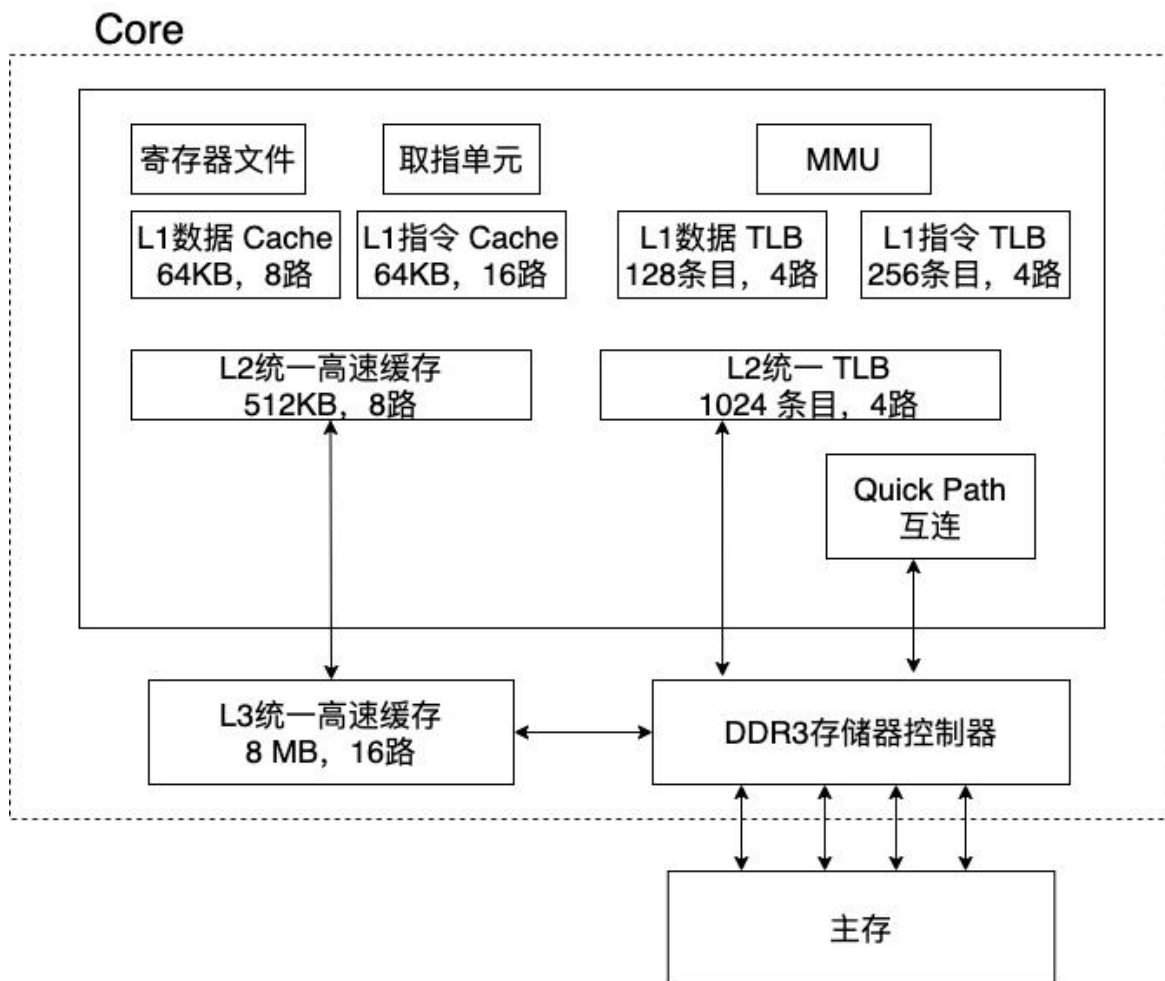
密 封

```

mov 0x200b3e(%rip),%eax #601044 <val>
add %edx,%eax
mov %eax,0x200b36(%rip) #601044 <val>
addl $0x1,-0x4(%rbp)
400512:
cmpl $0x3,-0x4(%rbp)#④
jl 4004f4 <sum+0xd>#⑤
mov 0x200b26(%rip),%eax # 601044 <val>
pop %rbp
Retq

```

16. 某处理器的虚拟地址为 32 位。虚拟内存的页大小是 4KB，物理地址为 48 位，Cache 块大小为 32B。物理内存按照字节寻址。其内部结构如下图所示，依据这个结构，回答下面几个问题。（12 分）



(1) L1 数据 Cache 有多少组，相应的 Tag 位，组索引位和块内偏移位分别是多少？

(2) 对于某数据，其访问的虚拟地址为  $0x829358B$ ，则该地址对应的 VPO 为多少？对应的 L1 TLBI 位为多少？（用 16 进制表示）

(3) 对于某指令，其访问的物理地址为  $0x829358B$ ，则该地址访问 L1 Cache 时，CT 位为多少？CO 位为多少？（用 16 进制表示）

17. 两个 C 语言程序 main2.c、addvec.c 如下所示:

|                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>/* main2.c */ /* \$begin main2 */ #include &lt;stdio.h&gt; #include "vector.h"  int x[2] = {1, 2}; int y[2] = {3, 4}; int z[2];  int main() {     addvec(x, y, z, 2);     printf("z = [%d %d]\n", z[0], z[1]);     return 0; } /* \$end main2 */</pre> | <pre>/* addvec.c */ /* \$begin addvec */ int addcnt = 0;  void addvec(int *x, int *y,             int *z, int n) {     int i;      addcnt++;      for (i = 0; i &lt; n; i++)         z[i] = x[i] + y[i]; } /* \$end addvec */</pre> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

用如下两条指令编译、链接，生成可执行程序 prog2:

gcc -m64 -no-pie -fno-PIC -c addvec.c main2.c

gcc -m64 -no-pie -fno-PIC -o prog2 addvec.o main2.o

运行指令 objdump -dxs main2.o 输出的部分内容如下:

Disassembly of section .text:

0000000000000000 <main>:

```
0: 48 83 ec 08 sub $0x8,%rsp
4: b9 02 00 00 00 mov $0x2,%ecx
9: ba 00 00 00 00 mov $0x0,%edx
 a: R_X86_64_32 z
e: be 00 00 00 00 mov $0x0,%esi
 f: R_X86_64_32 y
13: bf 00 00 00 00 mov $0x0,%edi
 14: R_X86_64_32 x
18: e8 00 00 00 00 callq 1d <main+0x1d>
 19: R_X86_64_PC32 addvec-0x4
1d: 8b 0d 00 00 00 00 mov 0x0(%rip),%ecx # 23 <main+0x23>
 1f: R_X86_64_PC32 z
23: 8b 15 00 00 00 00 mov 0x0(%rip),%edx # 29 <main+0x29>
 25: R_X86_64_PC32 z-0x4
29: be 00 00 00 00 mov $0x0,%esi
 2a: R_X86_64_32 .rodata.str1.1
2e: bf 01 00 00 00 mov $0x1,%edi
33: b8 00 00 00 00 mov $0x0,%eax
38: e8 00 00 00 00 callq 3d <main+0x3d>
 39: R_X86_64_PC32 __printf_chk-0x4
3d: b8 00 00 00 00 mov $0x0,%eax
42: 48 83 c4 08 add $0x8,%rsp
46: c3 retq
```

objdump -dxs prog2 输出的部分内容如下 (■是没有显示的隐藏内容):

SYMBOL TABLE:

```
0000000000400238 l d .interp 0000000000000000 .interp
0000000000400254 l d .note.ABI-tag
0000000000000000 l df *ABS* 0000000000000000 main2.c
0000000000601038 g *ABS*0000000000000000 _edata
000000000060103c g O .bss 0000000000000008 z
0000000000601030 g O .data 0000000000000008 x
0000000000000000 F *UND* 0000000000000000 addvec
0000000000601018 g .data 0000000000000000 _data_start
00000000004007e0 g O .rodata 0000000000000004 _IO_stdin_used
0000000000601028 g O .data 0000000000000008 y
00000000004006f0 g F .text 0000000000000047 main
```

00000000004005c0 <addvec@plt>:

```
4005c0: ff 25 42 0a 20 00 jmpq *0x200a42(%rip) # 601008
 <_GLOBAL_OFFSET_TABLE_+0x20>
4005c6: 68 01 00 00 00 pushq $0x1
4005cb: e9 d0 ff ff ff jmpq 4005a0 <_init+0x18>
```

00000000004005d0 <\_\_printf\_chk@plt>:

```
4005d0: ff 25 3a 0a 20 00 jmpq *0x200a3a(%rip) # 601010
 <_GLOBAL_OFFSET_TABLE_+0x28>
```

....

00000000004006f0 <main>:

```
4006f0: 48 83 ec 08 sub $0x8,%rsp
4006f4: b9 02 00 00 00 mov $0x2,%ecx
4006f9: ba ① _ _ _ _ mov ■■■■,%edx
4006fe: be ② _ _ _ _ mov ■■■■,%esi
400703: bf ③ _ _ _ _ mov ■■■■,%edi
400708: e8 ④ _ _ _ _ callq 4005c0 <addvec@plt>
40070d: 8b 0d ⑤ _ _ _ _ mov ■■■■(%rip),%ecx # 601040 <z+0x4>
400713: 8b 15 ⑥ _ _ _ _ mov ■■■■(%rip),%edx # 60103c <z>
400719: be e4 07 40 00 mov $0x4007e4,%esi
40071e: bf 01 00 00 00 mov $0x1,%edi
400723: b8 00 00 00 00 mov $0x0,%eax
400728: e8 ⑦ _ _ _ _ callq 4005d0 <__printf_chk@plt>
40072d: b8 00 00 00 00 mov $0x0,%eax
400732: 48 83 c4 08 add $0x8,%rsp
400736: c3 retq
```

1) 请指出 addvec.c main2.c 中哪些是全局符号? 哪些是强符号? 哪些是弱符号? 以及这些符号经链接后在哪个节? (5 分)

2) 根据上述信息, 思考 main 函数中空格①--⑦所在语句所引用符号的重定位结果是什么? 请以 16 进制 4 字节数值填写这些空格, 将机器指令补充完整 (写出任意 3 个即可)。(3 分)

## 五、综合设计题（共每题 10 分，共 30 分）

18. 请分别写出 Y86-64 CPU 顺序结构设计与实现中，call 指令和 ret 指令在各阶段的微操作(其中 call XXX 指令的取指阶段的微操作已经给出)。(12 分)

| 指令    | call XXX                                    | ret |
|-------|---------------------------------------------|-----|
| 取指    | icode:ifun $\leftarrow$ M <sub>1</sub> [PC] |     |
|       |                                             |     |
|       | valC $\leftarrow$ M <sub>8</sub> [PC+1]     |     |
|       | valP $\leftarrow$ PC+9                      |     |
| 译码    |                                             |     |
|       |                                             |     |
| 执行    |                                             |     |
| 访存    |                                             |     |
| 写回    |                                             |     |
| 更新 PC |                                             |     |

19. 优化矩阵乘法的程序:矩阵  $c[n,n] = a[n,n] * b[n,n]$

```
void MatrixMult(double *a ,double *b,double *c,int n){
 for(int i=0;i<n;i++)
 for(int j=0;j<n;j++)
 {
 c[i][j]=0;
 for(int k=0; k<n;k++)
 c[i][j]+=a[i][k]*b[k][j];
 }
}
```

请针对该程序进行面向编译的优化、面向 CPU 的优化（提高指令的并行性），本题不考虑面向 cache 的优化，写出优化后的程序，并说明优化的依据。（8 分）