计算机组成原理部分试题

> 注1：以下部分题目的答题量远超真题，在408答题卡上可能写不下，可以在草稿纸或笔记本上答题
>
> 注2：可以先跳到第5题，从第5题开始做。今晚直播主要探讨操作系统部分的试题。

1.某字长为8位的计算机中，带符号整数采用补码表示，a=-68,b=-80。a和b分别存放在寄存器A和B中（可用 A_7 ~ A_0 表示寄存器A的最高位到最低位，其他寄存器类似），请问：

（1）寄存器A和B中的内容分别为?

（2）若a+b后的结果存放在寄存器C中，则寄存器C的内容是什么？运算结果是否正确？此时，符号标志SF、溢出标志OF、和零标志ZF各是什么？加法器最高位进位F_{out}是什么？

（3）若a-b后的结果存放在寄存器D中，则寄存器D的内容是什么？运算结果是否正确？此时，符号标志SF、溢出标志OF、和零标志ZF各是什么？加法器最高位进位F_{out}是什么？

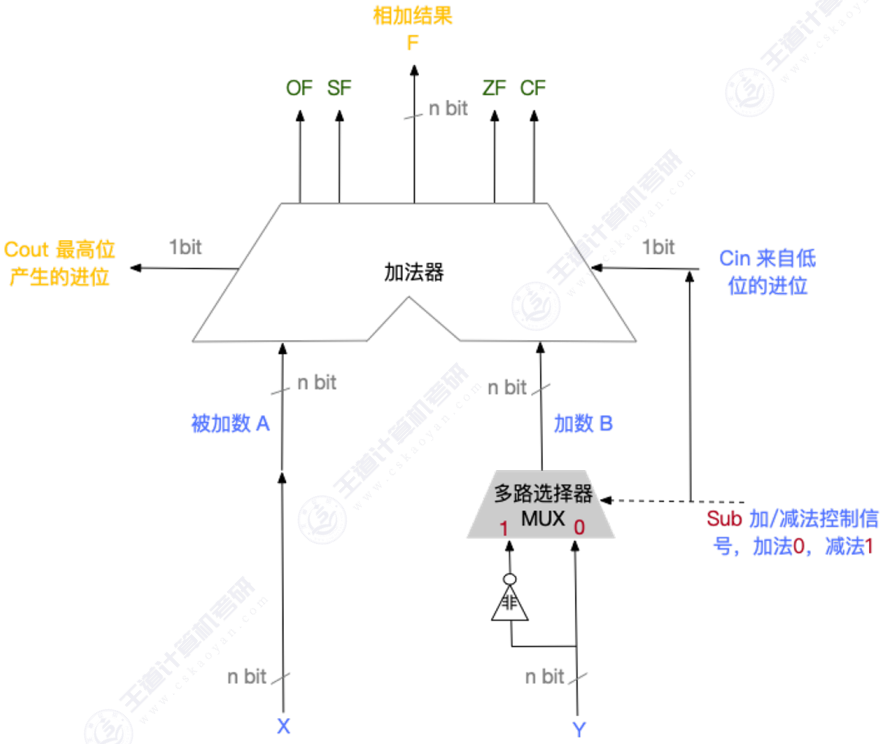
（4）请画出 8bit 带标志位的补码加法器的示意图，结合示意图，说明该加法器是如何实现减法运算的?

（5）结合（4）的图示，给出 OF、SF、ZF、CF 的逻辑表达式，并说明每个标志位的含义和作用。

（6）在C语言中，我们常使用 if-else 语句实现程序的分支结构。①如果 a、b 是带符号整数，那么计算机硬件是如何判断条件 if(a>=b) 是否满足的？②如果 a、b 是无符号整数，那么计算机硬件是如何判断条件 if(a>=b) 是否满足的?

> 【参考答案】
>
> （1）[–68]补=[–1000100]补=1011 1100B=BCH。[–80]补=[–1010000]补=1011 0000B=B0H。所以，寄存器A和B中的内容分别是BCH和B0H。
>
> （2）[x+y]补=[x]补+[y]补=1011 1100 + 1011 0000 = (1) 0110 1100 = 6CH，最高位前面的一位1被丢弃，因此，寄存器C中的内容为6CH。对应的真值为+108，结果不正确。根据运算结果可知：
>
> - SF=0。表示运算结果为正数。
> - OF=1。表示带符号数加法运算发生溢出。
> - ZF=0。表示运算结果不为全0
> - 加法器最高位产生的进位F_{out}=1。
>
> （3）[x–y]补=[x]补+[–y]补=1011 1100 + 0101 0000 = (1) 0000 1100 = 0CH，最高位前面的一位1被丢弃，因此，寄存器D中的内容为0CH，对应的真值为+12，结果正确。根据运算结果可知：
>
> - SF=0。表示运算结果为正数。
> - OF=0。表示带符号数减法运算未溢出。
> - ZF=0。表示运算结果不为全0。
> - 加法器最高位产生的进位F_{out}=1。

（4）可参考基础课课件。



当进行减法运算 X-Y 时，需要将Y全部位按位取反末位加1，用加法等价实现减法，即 X-Y 等价于 X+(Y全部位按位取反末位加1)。

如上图所示，将被减数X输入到加法器的一端。控制信号 sub=1，此时减数Y通过非门实现全部位按位取反，并通过多路选择器输入到加法器的另一端。同时，控制信号 sub=1 作为加法器的进位输入，从而实现“末位加1”。这样就用加法器等价实现了减法运算。

（5）可参考基础课课件。四个标志位的生成、作用非常重要，一定要高度重视！

- 溢出标志为 OF 的含义是“带符号数的运算结果是否溢出”，OF=1表示有溢出，OF=0表示没有溢出。注意：OF对无符号数的运算无意义。加/减法运算中常用两种方法确定OF的值：
  - 方法1：若两个加数的符号位相同，但与结果的符号位相异，则溢出。逻辑表达式如下

$$OF = \bar{A}_7\bar{B}_7C_7 + A_7B_7\bar{C}_7$$

  - 方法2：若最高位上的进位和次高位上的进位不同，则溢出。逻辑表达式如下：

$$OF = F_{out} \oplus \text{次高位产生的进位}$$

- 进位/借位标志 CF 的含义是“无符号数加法/减法运算是否发生了进位/借位”。当CF=1时说明加法/减法发生了进位/借位，同时也说明发生溢出；当CF=0时说明加法/减法没有发生进位/借位，同时也说明没有发生溢出。逻辑表达式为：$CF=F_{out}\oplus Sub$（Sub=1表示减法，Sub=0表示加法）。注意：CF对带符号数的运算无意义。
- 符号标志 SF 的含义是“有符号数的运算结果的正负性”，SF=0表示结果为正，SF=1表示结果为负。逻辑表达式为 $SF=C_7$。注意：SF对无符号数的运算无意义。

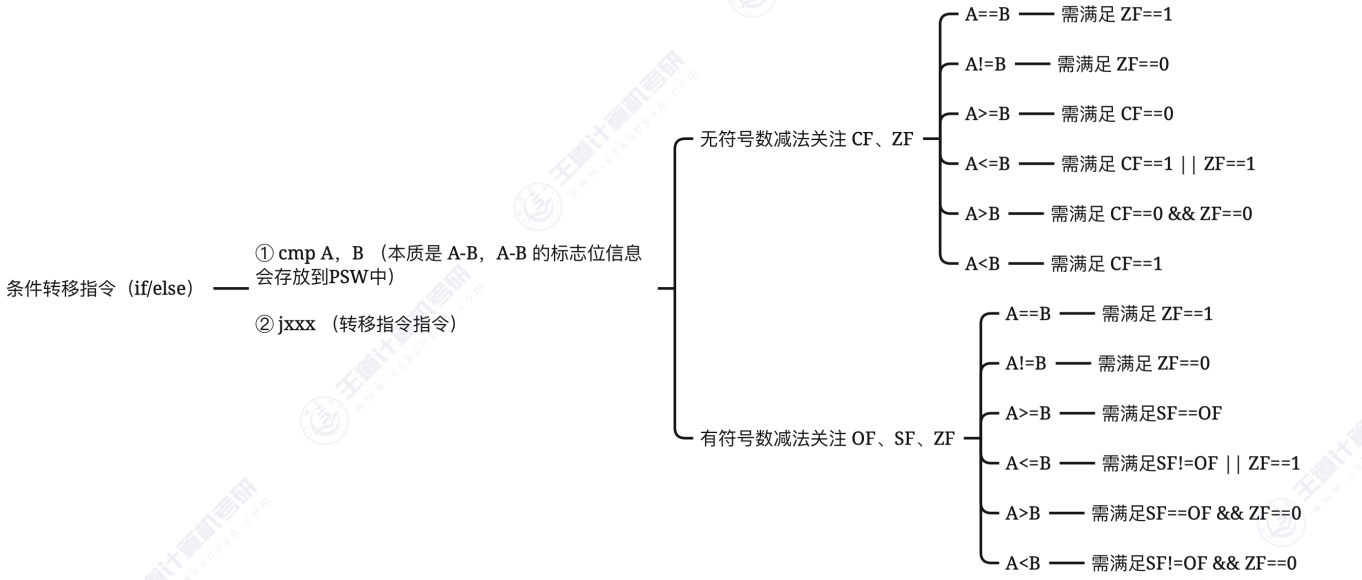
- 零标志 ZF 的含义是“运算结果是否为0”。当ZF=0时表示运算结果非0；当ZF=1时表示运算结果为0。逻辑表达式为：

$$ZF = C_7 + C_6 + C_5 + C_4 \bar{+} C_3 + C_2 + C_1 + C_0$$

注1：上面公式显示有bug，应该是C7~C0全部或非。

注2：ZF标志位对带符号数、无符号数的运算都有意义。

（6）参考计组强化直播P4思维导图：



2.有如下C语言过程：

```
int sum(int n)
{
  int i=1;
  int result=0;
  for (i=1;i<=n;i++)
    result+=i;
  result*=2;
  return result;
}
```

对应的部分汇编代码为：

```
1       movl    8(%ebp),%ecx
2       movl    $0,%eax
3       movl    $1,%edx
4       cmpl    %ecx,%edx
5       jg      L2
6   L1: addl    %edx,%eax
7       addl    $1,%edx
8       cmpl    %ecx,%edx
9       jle     L1
10  L2: sal     $1,%eax
11      ret
```

请问：

（1）该过程计算结果为?

（2）eax、edx、ecx分别用于保存哪个变量？返回值在哪个寄存器中?

（3）请解释该汇编语言中，如何实现 for 循环?

（4）第10条汇编指令 sal 实现了算数左移，该指令对应C语言中哪句代码？还可以用别的指令实现同样的功能吗?

（5）哪几条指令会产生程序执行流的转移?

【参考答案】

（1）result = n*(n+1)。（注：如果n的值太大，那么可能会溢出，返回结果可能出错）

（2）由第2条指令知eax初值为0，即eax是result

由第3条指令知edx初值为1，即edx是i

由第8条指令知i与**ecx**做比较，可知**ecx**是n

返回result值，在eax中

（3）参考：

```
1       movl    8(%ebp),%ecx        #将参数n的值从栈中取出，放入寄存器 ecx
2       movl    $0,%eax             #变量 result 初始值为0，存在寄存器 eax
3       movl    $1,%edx             #变量 i 初始值为1，存在寄存器 edx
4       cmpl    %ecx,%edx           #比较 i和n，本质上是在做 i-n。AT&T汇编语言，被减数在后，减
数在前
5       jg      L2                  #如果 i 比 n 大，则跳转到 L2
6   L1: addl    %edx,%eax           #实现 result = result + i
7       addl    $1,%edx             #实现 i++
8       cmpl    %ecx,%edx           #比较 i和n
9       jle     L1                  #如果 i比n小，或i=n，则跳转到 L1，否则顺序执行下一句
10  L2: sal     $1,%eax             #算数左移两位，实现 result = result*2
11      ret                         #返回上一层函数
```

> （4）sal 指令对应 result*=2;
>
> 还可以用加法指令，实现 (eax)+(eax)→eax
>
> 也可以用乘法指令，实现 (eax)×2→eax
>
> （5）第5、9行，条件转移指令，有可能改变PC的值；第11行 ret 指令，一定会改变PC的值。
>
> 注意：本题采用AT&T格式的x86汇编语言，请大家复习 AT&T 和 intel 格式的区别（详见基础课），以防考试中突然采用 AT&T 格式导致看不懂题目。

3.某程序中有如下循环代码段P：“for(int i = 0; i < N; i++) sum+=A[i];”。假设编译时变量sum和i分别分配在寄存器R1和R2中。常量N在寄存器R6中，数组A的首地址在寄存器R3中。程序段P起始地址为0804 8100H，对应的汇编代码和机器代码如下表所示。

编号	地址	机器代码	汇编代码	注释
1	08048100H	00022080H	loop: sll R4, R2, 2	(R2)<<2 → R4
2	08048104H	00083020H	add R4, R4, R3	(R4) + (R3) → R4
3	08048108H	8C850000H	load R5, 0(R4)	((R4) + 0) → R5
4	0804810CH	00250820H	add R1, R1, R5	(R1) + (R5) → R1
5	08048110H	20420001H	add R2, R2, 1	(R2) + 1 → R2
6	08048114H	1446FFFAH	bne R2, R6, loop	if(R2)!=(R6) goto loop

执行上述代码的计算机M采用32位定长指令字，其中分支指令bne采用如下格式：

31 26	25 21	20 16	15 0
OP	Rs	Rd	OFFSET

OP为操作码；Rs和Rd为寄存器编号；OFFSET为偏移量，用补码表示。请回答下列问题，并说明理由。

1）M的存储器编址单位是什么？

2）已知sll指令实现左移功能，数组A中每个元素占多少位？

3）表中bne指令的OFFSET字段的值是多少？已知bne指令采用相对寻址方式，当前PC内容为bne指令地址，通过分析表中指令地址和bne指令内容，推断出bne指令的转移目标地址计算公式。

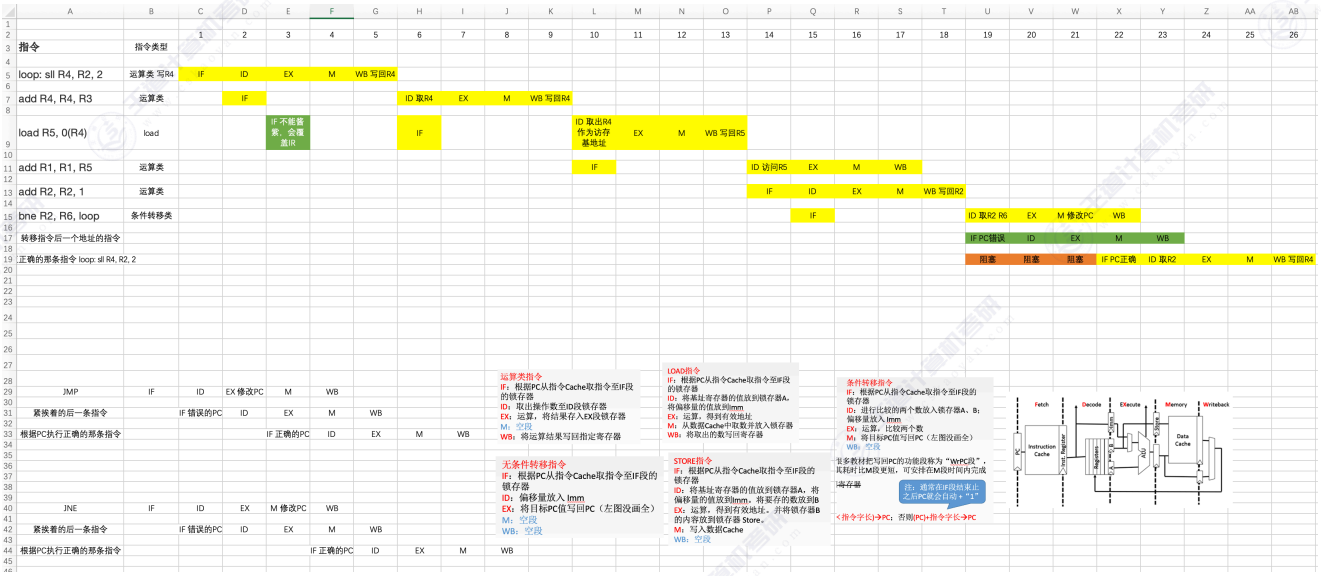
4）若M采用如下“按序发射、按序完成”的5级指令流水线：IF（取值）、ID（译码及取数）、EXE（执行）、MEM（访存）、WB（写回寄存器），且硬件不采取任何转发措施，分支指令的执行均引起3个时钟周期的阻塞，则P中哪些指令的执行会由于数据相关而发生流水线阻塞？哪条指令的执行会发生控制冒险？为什么指令1的执行不会因为与指令5的数据相关而发生阻塞？

5）模仿下图画出上述6条指令的指令流水线。

	时间单元													
指令	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I_1	IF	ID	EX	M	WB									
I_2		IF	ID	EX	M	WB								
I_3			IF				ID	EX	M	WB				
I_4							IF				ID	EX	M	WB

【参考答案】

这个题目是2014年真题，前四个小问就是真题本题，可参考2023王道书5.6.6大题6。第五个小问是一个扩展，参考答案如下：



4.假定某输入设备A的速度为 n B/s，对应I/O接口中有一个 32位数据缓冲寄存器，请回答下列问题，并给出计算过程。

（1）若设备A采用程序查询I/O方式，CPU每隔 t1 会进行一次程序查询，每次程序查询时间开销为 t2（两次程序查询之间的时间间隔是 t1，也就是说 t2 是 t1 的子区间），为了使设备A的输入数据不丢失，需要满足什么条件？

（2）若设备A采用中断控制方式，中断处理的时间总开销为 t3，则结合（1）中的条件，要让中断控制方式的效率比程序查询方式的效率更高，需要满足什么条件？

（3）若设备A采用DMA控制方式，DMA预处理和后处理的时间总开销为 t4，DMA传送块大小为 d字节，假设DMA与CPU之间没有访存冲突，则要让DMA控制方式的效率比中断控制方式的效率更高，需要满足什么条件?

> 【参考答案】
>
> （1）I/O接口的缓冲区大小 32bit = 4B 数据。而设备的输入速度是 n B/s，因此每隔 4/n 秒，缓冲区就会被充满，CPU要及时将数据取走，否则就可能导致数据丢失。因此需要满足 t1 < 4/n
>
> （2）首先要理解什么叫“中断控制方式的效率更高”——就是采用中断控制方式的时候，CPU花在处理I/O上的时间比 程序查询方式更少。我们不妨计算 1秒之内，CPU花在处理I/O上的时间是多少。
>
> 若采用中断控制方式，则数据缓冲区每充满一次，就会产生一次中断，一秒之内产生中断的次数是 n/4 次，每次中断处理时间开销为 t3。因此，一秒之内，CPU处理中断所花时间为 (n/4)t3
>
> 若采用程序查询方式，一秒之内，CPU进行程序查询的次数为 1/t1，每次程序查询耗时 t2。因此，一秒之内，CPU程序查询所花时间为 (1/t1)t2
>
> 综上，要使中断控制方式效率更高，需要满足 (n/4)t3 < (1/t1)t2
>
> （3）要让 DMA控制方式 的效率更高，就是要保证采用 DMA方式时，CPU花在处理I/O上的时间更少。根据题目条件，我们不妨计算每传送 d 字节，CPU的时间开销。
>
> 若采用DMA控制方式，则每传送一块 d字节，CPU用于DMA预处理和后处理的时间总开销为 t4。
>
> 若采用中断控制方式，则数据缓冲区每充满一次，就会产生一次中断，传送 d 字节，共产生中断 d/4 次，每次中断处理时间开销为 t3。因此，每传送 d 字节，CPU用于中断处理的时间总开销为 (d/4)t3
>
> 综上，要使DMA控制方式效率更高，需要满足 t4 < (d/4)t3
>
> 通过这个题目，可以感受到，三种I/O方式没有绝对的孰优孰劣，要综合考虑设备速度、CPU处理一次IO的开销。

5.下面是 Intel Core i7 地址转换过程的简要示意图，已知该机按字节编址，请根据图示回答下面的问题：

1）该CPU支持的虚拟地址空间大小是多少？物理地址空间大小是多少?

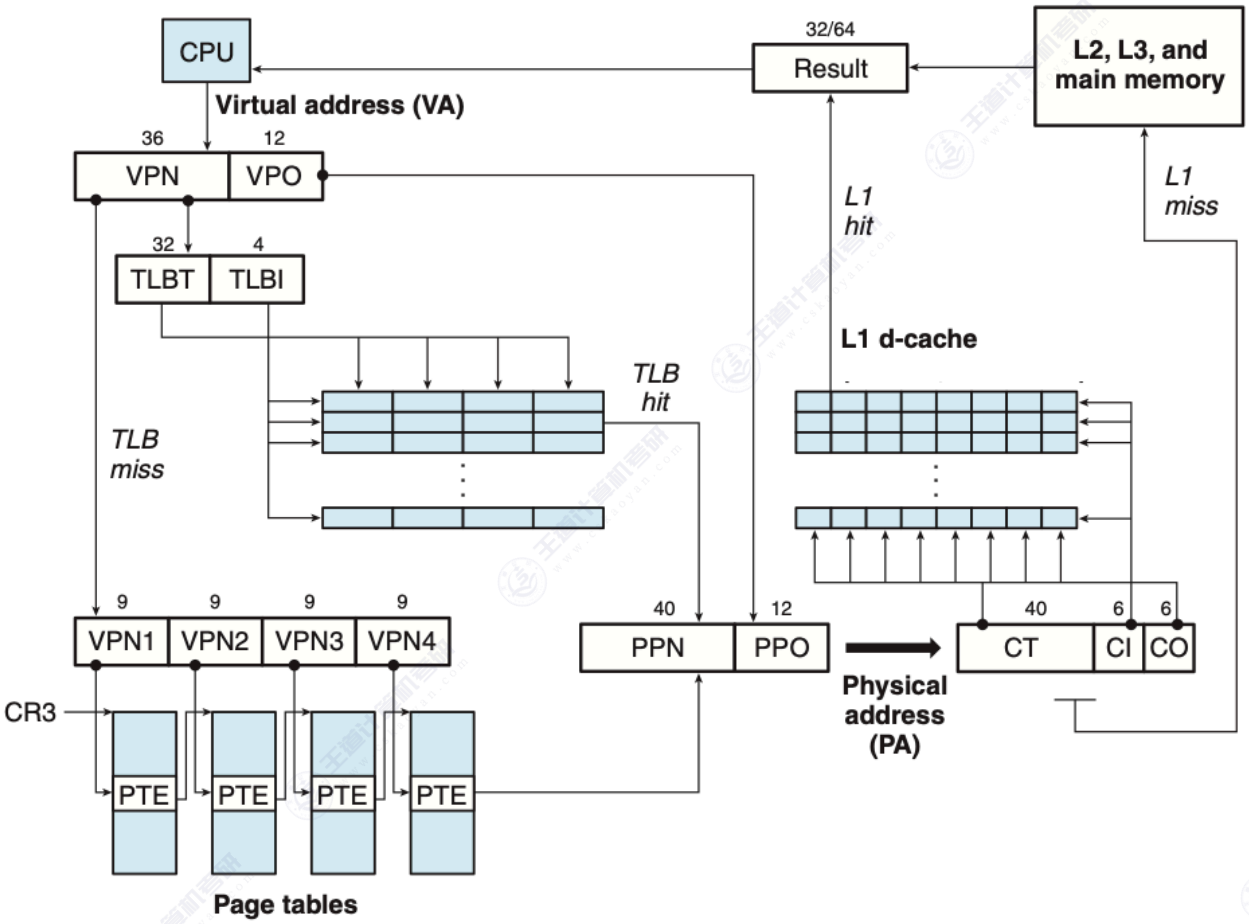
2）该处理器支持几级页表？每个页面的大小是多少？每个页表项的大小是多少?

3）TLB采用什么映射方式？图示中 TLBT、TLBI 的作用是什么？请描述从虚拟地址转化为物理地址的过程。

4）Cache采用什么映射方式？每个Cache行的数据大小是多少？图示中 CT、CI、CO 的作用是什么?

5）若Cache采用LRU替换策略，且采用写回法，请画出每个Cache行的结构，并计算图中 L1 d-Cache 的实际大小是多少?

6）请描述根据物理地址访问图中Cache的过程。



【参考答案】

1）看图可知，虚拟地址VA总位数为36+12=48bit，物理地址PA总位数为40+12=52bit。

因此，虚拟地址空间大小为 2^{48}B = 256TB，物理地址空间大小为 2^{52}B = 4PT。

注：复习一下大纲要求掌握的数据单位 K、M、G、T、P、E、Z。依次递增，每增长一级就是乘 2^{10}的关系。在表示浮点数运算速度时，各单位之间每增长一级就是乘 10^3。

2）看图可知，该处理器支持4级页表。每个页面大小是 2^{12}=4KB。通常来说采用多级页表时，最理想的情况是每一级的页表大小不超过一个页面，而从图中可知，每一级的页号占9位，也就是说每一级页表有 2^9 个表项，因此每个页表项的大小为 $2^{12}/2^9$ = 8B。

3）TLB采用组相联映射，具体来说，4路组相联映射，共2^4 = 16组。TLBT，即 TLB Tag，TLB标记信息。TLBI，即TLB Index，索引号，也就是组相联映射中的“组号”，从图中可知，TLBI共4bit，说明总共有 2^4 = 16组。

地址转化过程如下：

1. 查TLB
    1. 虚拟地址VA共48位，取出前36位作为虚页号VPN，VPN再拆分为 32位TLBT、4位TLBI。
    2. 用4位 TLBI 确定TLB组号，一个分组中包含 4 个表项
    3. 用32位TLBT 和分组中的 4个表项对比，TLBT匹配，且有效位为1，说明TLB命中，直接得到物

理页框号PPN（40bit）

2. 若TLB未命中，则查内存中的页表
    1. 虚拟地址的前36位作为虚页号VPN。再拆分为 9+9+9+9，也就是4级页号
    2. 根据每一级页号去查每一级页表。当然，查页表的过程有可能会缺页，如果缺页了还得调页
    3. 查到最后一级页表，就可以确定物理页框号PPN（40bit）
3. 将物理页框号PPN 和页内偏移量 VPO 拼接得到完整的物理地址 PA

4）Cache 采用组相联映射，具体来说，8路组相联映射，共 2^6=64组

每个Cache行的数据大小是 2^6=64B

40位CT为Cache Tag，用于判断Cache是否命中

6位CI为Cache Index，用于表示Cache组号。其位数可以反映出总共有几个分组

6位CO为Cache Offset，也就是Cache字块内地址（块内偏移量）。其位数可以反映出每个Cache字块大小

5）从图示可知，该Cache采用8路组相联映射，也就是说每个分组中有8个Cache行。若采用LRU替换策略，得用 3bit 来标记组内各个Cache行。采用写回法（write-back），需要 1bit 作为脏位。每个Cache行的结构如下：

Tag标记	有效位	LRU替换信息	脏位	数据
40bit	1bit	3bit	1bit	64B

图中的L1 d-cache，共 8路×64组 = 512 行，每行的大小为 (40+1+3+1)bit + 64B。所以实际大小是 512×[(40+1+3+1)bit + 64B] = 自己算吧┐(´∀`)┌

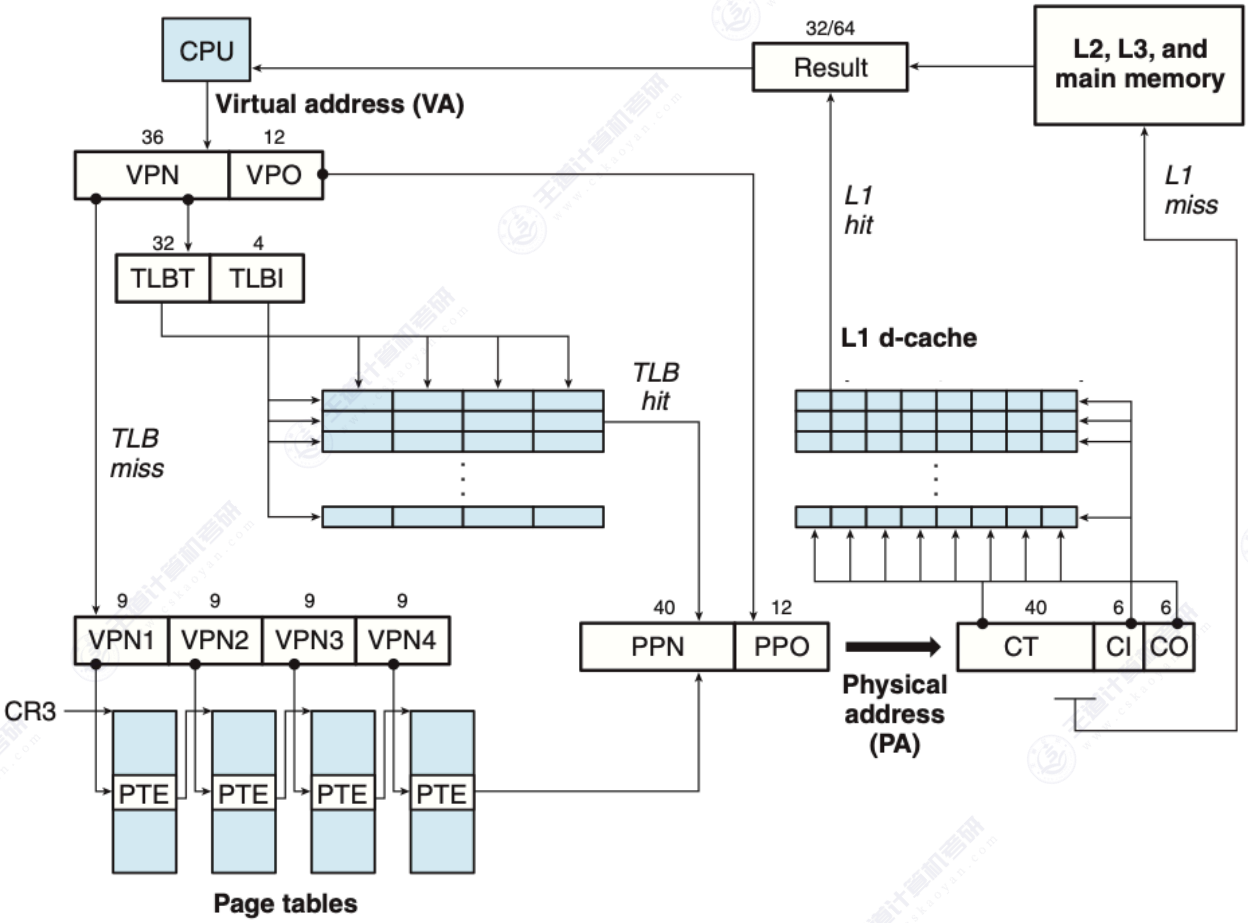
6）物理地址 PA 为40+12 = 52bit

- 就像图示那样，拆分为 40bit CT，6bit CI，6bit CO
- 根据 6bit CI 找到对应 Cache分组，分组内共8个Cache行
- 物理地址的前 40bit CT 与分组内的 8个Cache行的 Tag标记对比，如果Tag能匹配，且有效位为1，则Cache命中，找到一个Cache行
- 根据字块内偏移量，6bit CO，从Cache行的64B数据中，访问数据
- 如果Cache不命中，则需要根据物理地址PA去访存（当然，实际上在 i7 处理器中，还有 L2级Cache、L3级Cache，都没命中才会访问主存）

操作系统部分试题

> 注1：以下部分题目的答题量远超真题，在408答题卡上可能写不下，可以在草稿纸或笔记本上答题
>
> 注2：不考操作系统的同学可以跳过这部分试题

6.下面是 Intel Core i7 地址转换过程的简要示意图，已知该机按字节编址，请根据图示回答下面的问题：



1）该CPU支持的虚拟地址空间大小是多少？物理地址空间大小是多少？（同计组第5题）

2）该处理器支持几级页表？每个页面的大小是多少？每个页表项的大小是多少？（同计组第5题）

3）若TLB命中率为98%，TLB访问时间1ns，内存访问时间100ns。并假设当TLB miss时才开始访问内存中的页表，且各级页表常驻内存，则平均的地址转换时间是多少？

4）TLB访问时间1ns，内存访问时间100ns。并假设当TLB miss 时才开始访问内存中的页表，只有顶级页表常驻内存。查多级页表的过程中，下一级页表未调入内存的概率是 1%。已知处理一个缺页中断平均需要8ms。如果要满足平均地址转换时间小于2ns，那么TLB命中率需要至少多少？

5）地址转换完成后，得到目标物理地址。访问目标物理地址的过程中，Cache命中率为98%，Cache访问时间1ns，内存访问时间100ns。并假设访问 Cache 的同时就开始访问内存。当发生缺页时，若有一个可用的空页或被置换的页未被修改，则它处理一个缺页中断需要5ms；若被置换的页已被修改，则处理一缺页中断因增加写回外存时间而需要10ms。发生缺页时，60%的概率需要置换一个被修改的页面，为保证访问目标物理地址的平均时间不超过80ns，可接受的最大缺页中断率是多少？

> 【参考答案】

1）2）两个小问与第5题相同，不再赘述

3)若TLB命中，则可以直接完成地址转换，耗时1ns。

若TLB未命中，需要查询四级页表，每查一级页表需要一次访存，因此地址转换耗时=1+100+100+100+100=401ns。

综上，平均地址转换时间 = 1×98% + 401×2% = 9 ns

4)这个小问可以跳过，太难了，不用深究。

若TLB命中，则可以直接完成地址转换，记耗时A=1ns。设TLB命中的概率是 p。

若TLB未命中，需要查询四级页表，一级页表常驻内存。

- 如果二级页表未调入内存，则三、四级页表也一定未调入内存，查页表过程中累计发生三次缺页。则耗时B = (1ns+100ns+8ms)+(1ns+100ns +100ns+8ms)+(1ns+100ns +100ns+100ns+8ms)+(1ns+100ns +100ns+100ns+100ns)
- 如果二级页表原本已调入内存，此时三级页表未调入内存的概率是1%。如果三级页表未调入内存，则四级页表也一定未调入内存，查页表过程中累计发生两次缺页。则耗时C = (1ns+100ns +100ns+8ms)+(1ns+100ns +100ns+100ns+8ms)+(1ns+100ns +100ns+100ns+100ns)
- 如果三级页表原本已调入内存，则四级页表未调入内存的概率是1%。查页表过程中累计发生一次缺页。则耗时D = (1ns+100ns +100ns+100ns+8ms)+(1ns+100ns +100ns+100ns+100ns)
- 如果四级页表原本已调入内存，则查页表过程中不会发生缺页，则耗时E=(1ns+100ns +100ns+100ns+100ns)

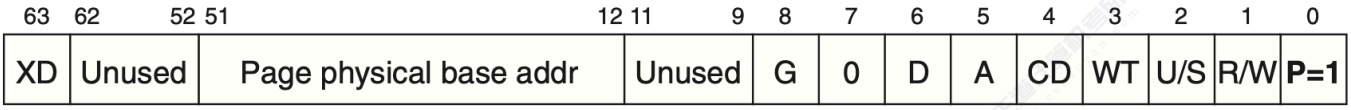
5)注意，这题设置了一个坑：“缺页”只可能在地址转换的过程中发生，只要能得到物理地址，就不可能再发生缺页。因此题目中给的都是干扰条件，这也是很多同学容易混淆错乱的部分。

地址转换完成后，访问目标物理地址时，只可能发生两种情况：

- ①Cache命中，访问物理地址耗时 1ns
- ②Cache未命中，则需要访问主存（不可能缺页），访问物理地址耗时 = 100 ns。注意：题干中已说明访问Cache的同时就开始访问主存，因此访问物理地址的耗时=100ns，而不是1+100ns。

7.下面这个图是 Intel core i7 页表项的结构，每个页表项 大小为8B。其中：

- P：有效位，或者翻译为存在位（Present），表示页面是否已经调入内存，p=0则缺页，p=1则不缺页。
- R/W：用 0/1 表示这个页面是否“只能读”，还是“可读可写”。比如存储常量的页面，就可以设置为只能读。
- U/S：用 0/1 表示这个页面在用户态下是否可访问，还是只有内核态下能访问。
- A：访问位，表示最近有没有被访问过，可以被置换算法使用。
- D：脏位，表示页面信息是否被修改过。当这个页面的数据被“写”过，就脏了。
- 第 12~51 位，共40bit，表示下一级页框号。



63	62	52	51	12	11	9	8	7	6	5	4	3	2	1	0
XD	Unused		Page physical base addr		Unused		G	0	D	A	CD	WT	U/S	R/W	**P=1**

回答下列问题：

1）结合第6题的条件，判断该CPU可支持的最大物理内存是多少?

2）当用户进程尝试访问操作系统内核区时，系统是如何处理这种非法访问行为的?

3）结合页表项示意图，简述操作系统内核是如何实现改进型 Clock 算法的?

4）第7个bit为0表示这一级页表大小为为 4KB；为1表示这一级的页表大小为4MB。但是，只有顶级页表才允许第7个bit为1，为什么?

> 【参考答案】
>
> 1. 2^{52}B
> 2. 操作系统内核区对应的页表项中，可通过 U/S 位为 0/1 表示这个页面是否允许在用户态下访问。系统在地址转换的过程中，通过当前CPU的状态位与 U/S 位来判断进程对内核区的访问是否合法。如果不合法，会触发异常。
> 3. 改进型Clock算法需要考虑进程驻留集中每个页表项的 A, D 两个位。
>
>   算法规则：将所有可能被置换的页面排成一个循环队列（本质是把对应的页表项排成一个循环队列）。
>
>   第一轮：从当前位置开始扫描到第一个（0, 0）的帧用于替换。本轮扫描不修改任何标志位
>
>   第二轮：若第一轮扫描失败，则重新扫描，查找第一个（0, 1）的帧用于替换。本轮将所有扫描过的帧访问位设为0
>
>   第三轮：若第二轮扫描失败，则重新扫描，查找第一个（0, 0）的帧用于替换。本轮扫描不修改任何标志位
>
>   第四轮：若第三轮扫描失败，则重新扫描，查找第一个（0, 1）的帧用于替换。
>
>   由于第二轮已将所有帧的访问位设为0，因此经过第三轮、第四轮扫描一定会有一个帧被选中，因此改进型CLOCK置换算法选择一个淘汰页面最多会进行四轮扫描优先。
> 4. 在多级页表的系统中，顶级页表常驻内存，也只有顶级页表的大小可以超过一个页面。

8.某男子足球俱乐部，有非常多的教练、队员。每次足球训练开始之前，教练、球员都需要先进入更衣室换衣服，可惜俱乐部只有一个更衣室。教练们脸皮薄，无法接受和别人共用更衣室。队员们脸皮厚，可以和其他队员一起使用更衣室。更衣完成后，才可以参加足球训练。

1）起初，俱乐部规定：如果队员和教练都要使用更衣室，则应该让教练优先。请使用P、V操作描述上述过程的互斥与同步，并说明所用信号量及初值的含义。

2）后来，因很多队员无法忍受教练优先的特权现象，俱乐部更改了规定——为体现“人人平等”的团队作风，队员和教练应遵循“先到先用”的原则，公平地使用更衣室。请使用P、V操作描述上述过程的互斥与同步，并说明所用信号量及初值的含义。

【参考答案】本题中，教练就是写者，队员就是读者，不过是读者写者问题换了个马甲而已。

1）按照题目要求，要求实现“写优先”，这种方法可能导致读者饥饿。“写优先”代码如下：

```
semophore readLock=1;                  //互斥信号量，用于给读者“上锁”
semophore writeLock=1;                 //互斥信号量，用于给写者“上锁”
semophore rmutex=1;                 //互斥信号量，实现对readCount的互斥访问
semophore wmutex=1;                 //互斥信号量，实现对writeCount的互斥访问
int readCount=0, writeCount=0;   //读者、写者的数量

//读者进程（在这个题里就是可以多人一起共用更衣室的队员们）
Reader(){
  while(1){
    P(readLock);          //每个读者到达时先对 read 上锁
    P(rmutex);
      readCount++;
      if(readCount==1) P(writeLock);   //第一个开始读的读者对写者上锁
    V(rmutex);
    V(readLock);          //每个读者正式开始读之前对 read 解锁
    队员使用更衣室;     //读者读文件
    P(rmutex);
      readCount－－;
      if(readCount==0) V(writeLock);   //最后一个读完的读者对写者解锁
    V(rmutex);
    队员参加足球训练;
  }
}

//写者进程（在这个题目里，对应必须独享更衣室的教练们）
Writer(){
  while(1){
    P(wmutex);
      writeCount++;
      if(writeCount==1) P(readLock);   //第一个到达的写者对读者上锁，这一步是实现“写优先”的
关键
    V(wmutex);
    P(writeLock);    //每个写者开始写之前都要对其他写者上锁，保证写者之间互斥
    教练使用更衣室;     //写者写文件
    V(writeLock);
    P(wmutex);
        writeCount－－;
        if(writeCount==0) V(readLock);   //最后一个写者写完之后，对读者解锁
    V(wmutex);
    教练参加足球训练;
  }
}
```

为了方便大家对比学习，下面再附上“读者优先”的实现，也就是王道书上的第一种实现方法。这种方式可能导致写者饥饿：

```
semaphore lock=1; //用于实现对共享文件的互斥访问
int count = 0;    //记录当前有几个读进程在访问文件
semaphore mutex = 1;  //用于保证对count变量的互斥访问

writer (){
  while(1){
    P(lock);    //写之前“加锁”
    写文件…
    V(lock);    //写完了“解锁”
  }
}

reader (){
  while(1){
    P(mutex); //各读进程互斥访问count
      if(count==0)  P(lock);  //第一个读者，读之前“上锁”
      count++;  //访问文件的读进程数+1
    V(mutex);
    读文件…
    P(mutex); //各读进程互斥访问count
      count--;  //访问文件的读进程数-1
      if(count==0)  V(lock);  //由最后一个读进程负责“解锁”
    V(mutex);
  }
}
```

2）第二小问要求实现“读写公平法”。也就是王道书里的第二种方法。下面是实现代码：

```
semaphore lock=1;          //用于实现对共享文件的互斥访问
int count = 0;           //记录当前有几个读进程在访问文件
semaphore mutex = 1;     //用于保证对count变量的互斥访问
semaphore queue = 1;     //用于实现“读写公平”。咸鱼注：可以将queue理解为一个“队列”，当资源暂不可访问时，无论读者、写者都需要公平排队

writer (){
  while(1){
    P(queue); //先排队
    P(lock);    //尝试“上锁”
    V(queue); //唤醒下一个队头进程
    写文件…
    V(lock);    //使用完资源，解锁
  }
}
```

```
reader (){
  while(1){
    P(queue);      //先排队
    P(mutex);      //互斥访问count
    if(count==0)
        P(lock);     //第一个到达的读者尝试“上锁”
    count++;         //读者计数+1
    V(mutex);
    V(queue);      //唤醒队头进程
    读文件…
    P(mutex);      //互斥访问count
    count--;         //读者计数-1
    if(count==0)     //最后一个离开的读者，负责“解锁”
        V(lock);
    V(mutex);
  }
}
```

对于上面三种解法，如果弄不清楚它们之间的区别，不妨带入一个例子看看。假设每个读者的读操作都耗时较长，读者写者到达的顺序是：

读者1——读者2——读者3——写者A——读者4——写者B——读者5

如果采用“写优先”的实现方法，那情况是这样的：读者1到达并开始读，紧接着读者2、读者3到达，都可以开始读；写者A到达，暂时不能写；读者4到达，暂时不能读；写者B到达，暂时不能写；读者5到达，暂时不能读；等读者1、2、3都读完之后，写者A开始写；写者A写完之后写者B开始写；写者B写完后读者4开始读，同时读者5也可以开始读。

如果采用“读者优先”的实现方法，那情况是这样的：读者1到达并开始读，紧接着读者2、读者3到达，都可以开始读；写者A到达，暂时不能写；读者4到达，可以开始读；写者B到达，暂时不能写；读者5到达，可以直接开始读；等读者1、2、3、4、5都读完之后，写者A、写者B才可以依次进行写。

如果采用“读写公平法”的实现方法，那情况是这样的：读者1到达并开始读，紧接着读者2、读者3到达，都可以开始读；写者A到达，暂时不能写；读者4到达，暂时不能读；写者B到达，暂时不能写；读者5到达，暂时不能读；等读者1、2、3都读完之后，写者A开始写；写者A写完之后读者4开始读；读者4读完后写者B开始写；写者B写完后读者5开始读。

9.俗话说，“干饭人，干饭魂，干饭人吃饭得用盆”。一荤、一素、一汤、一米饭，是每个干饭人的标配。饭点到了，很多干饭人奔向食堂。每个干饭人进入食堂后，需要做这些事：拿一个盆打荤菜，再拿一个盆打素菜，再拿一个盆打汤，再拿一个盆打饭，然后找一个座位坐下干饭，干完饭把盆还给食堂，然后跑路。现在，食堂里共有N个盆，M个座位。请使用P、V操作描述上述过程的互斥与同步，并说明所用信号量及初值的含义。

参考答案：显然，这个题目的关键是不能发生死锁。死锁问题应该参考哲学家问题的解决思路。在哲学家进餐问题中，我们解决死锁的方法有三种：

1. 奇数号哲学家必须先拿左手的筷子，偶数号哲学家必须先拿右手的筷子

2. 限制“最多允许4个哲学家同时进餐”
3. 仅当一个哲学家左右两边的筷子都可用时才允许哲学家拿筷子

其中，第一种方法的思想很难迁移到其他题目中。但是第二、第三种思想可以迁移到大多数死锁题目。

我们先来看一个标准的错误解法：

```
semaphore pot=N;  //同步信号量，用于表示“盆”资源，食堂里总共有N个盆
semaphore seat=M; //同步信号量，用于表示“作为”资源，食堂里总共有M个座位

//干饭人进程
EatMan(){
  进食堂;
  P(pot);   //拿一个盆
  打荤菜;
  P(pot);   //拿一个盆
  打素材;
  P(pot);   //拿一个盆
  打汤;
  P(pot);   //拿一个盆
  打饭;
  P(seat);  //占个座
  干饭;
  V(seat);  //让出座位
  V(pot);   //归还干饭盆
  V(pot);
  V(pot);
  V(pot);
  离开食堂;
}
```

显然，上面这种解法会导致死锁。假设同时来了好多个干饭人，每个人都拿三个盆，盆很快就会被拿光。那所有人都无法得到第四个盆，就会发生死锁。

下面我们模仿哲学家进餐问题的第二种解决思路。在哲学家问题中，共有5个哲学家，如果我们限制“最多允许4个哲学家同时进餐”，那么至少会有一个哲学家可以同时获得左右两只筷子，并顺利进餐，从而预防了死锁。

同样的思路可以迁移到干饭人问题中。每个干饭人需要同时持有4个盆才能干饭，那么最糟糕的情况是每个干饭人都持有3个盆，同时在等待第四个盆。此时，但凡再多一个盆，就至少能有一个干饭人可以顺利干饭，就不会死锁。因此我们可以限制同时抢盆的人数为 x，那么只要满足 3x + 1 ≤ N，则一定不会发生死锁，可得 x ≤ (N-1)/3。参考代码如下：

```
semaphore pot=N;      //同步信号量，用于表示“盆”资源，食堂里总共有N个盆
semaphore seat=M;     //同步信号量，用于表示“作为”资源，食堂里总共有M个座位
semaphore x=(N-1)/3;  //同步信号量x，用于表示最多允许多少个人同时干饭。(N-1)/3 向下取整
//干饭人进程
```

```
EatMan(){
  P(x);      //进食堂拿盆之前，先看看是否已到达人数上限
  进食堂;
  P(pot);    //拿一个盆
  打荤菜;
  P(pot);    //拿一个盆
  打素材;
  P(pot);    //拿一个盆
  打汤;
  P(pot);    //拿一个盆
  打饭;
  P(seat);   //占个座
  干饭;
  V(seat);   //让出座位
  V(pot);    //归还干饭盆
  V(pot);
  V(pot);
  V(pot);
  V(x);
  离开食堂;
}
```

上面这种做法，限制了人数上限，且先拿盆，再占座，一定不会发生死锁。当然，如果先占座、后拿盆，也不会死锁。事实上，如果座位的数量满足 seat ≤ (N-1)/3，那么甚至可以不设置专门的信号量x，完全可以先占座，后拿盆，也一定不会死锁。因为座位的数量就可以限制同时抢盆的人数。

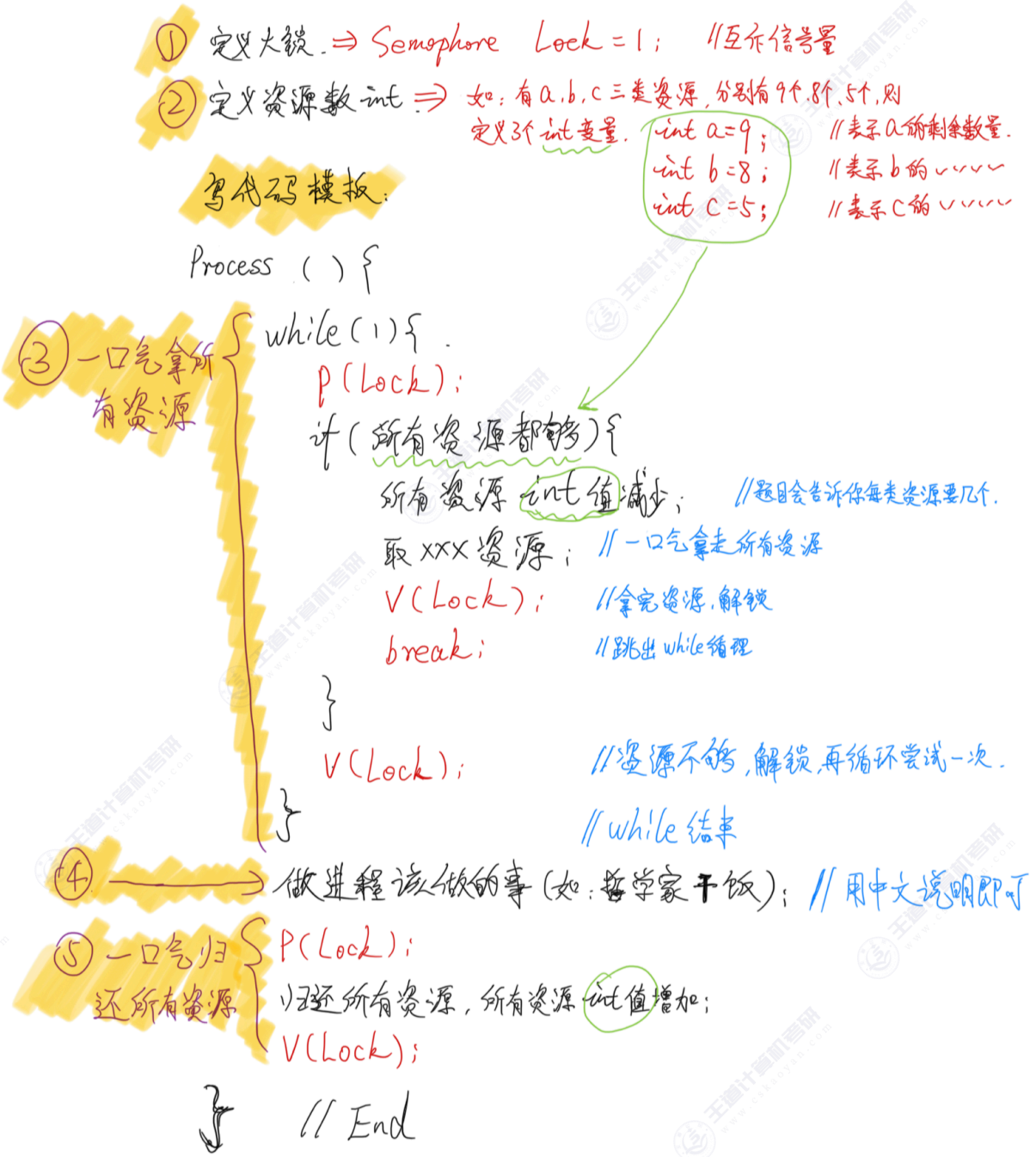
下面我们再模仿哲学家问题的第三种解决思路——仅当一个哲学家左右两边的筷子都可用时才允许哲学家拿筷子。破坏了“请求和保持”条件，采用“静态分配”的思想，让进程一口气获得所有资源，再开始运行。代码如下：

```
semaphore mutex=1;     //互斥信号量，保证所有进程对 pot 变量、seat 变量的访问是互斥的。
semaphore pot=N;       //用于表示“盆”资源，食堂里总共有N个盆
semaphore seat=M;      //用于表示“作为”资源，食堂里总共有M个座位

//干饭人进程
EatMan(){
  进食堂;
  P(mutex);
  P(pot);    //一口气拿四个盆，并占座
  P(pot);
  P(pot);
  P(pot);
  P(seat);
  V(mutex);

  打荤菜;
  打素材;
```

```
    打汤;
    打饭;
    干饭;

    V(seat);  //让出座位
    V(pot);   //归还干饭盆
    V(pot);
    V(pot);
    V(pot);
    离开食堂;
}
```

这个题目想告诉大家的是，哲学家进餐问题的解决思路中，后两种方法更为通用，可以作为考试时主要的策略。大家再思考一下，限制人数上限、一口气拿所有资源，哪种方案的并发度更高一些呢？显然是后者对吧。“限制人数上限”的方案中，最糟糕的情况是，只有一个人获得了4个盆，其余进程都只有3个盆，也就是说只有1个进程可以顺利运行下去，因此并发度低。

最后，总结一个“使用int变量表示资源”的哲学家进餐问题解题模板，使用下面这种方法解决哲学家进餐问题，可以保证进程间的并发度最高，但缺点是进程会陷入“忙等”：




```
semaphore mutex=1;       //互斥信号量，保证所有进程对 pot 变量、seat 变量的访问是互斥的。
int pot=N;               //用于表示“盆”资源，食堂里总共有N个盆
int seat=M;              //用于表示“作为”资源，食堂里总共有M个座位

//干饭人进程
EatMan(){
  进食堂;

  while(1){
    P(mutex);    //资源上锁
    if ( pot>=4 && seat>=1 ) {   //所有资源都够
      pot -=4;   //一口气拿走所有资源
      seat -=1;
      V(mutex); //拿完资源，解锁
```

```
        break;      //拿完资源，跳出循环
      }
      V(mutex);     //资源不够，解锁
    } //while

    打荤菜;
    打素材;
    打汤;
    打饭;
    干饭;

    P(mutex);     //资源上锁
    pot +=4;      //一口气归还所有资源
    seat +=1;
    V(mutex);     //资源解锁

    离开食堂;
}
```

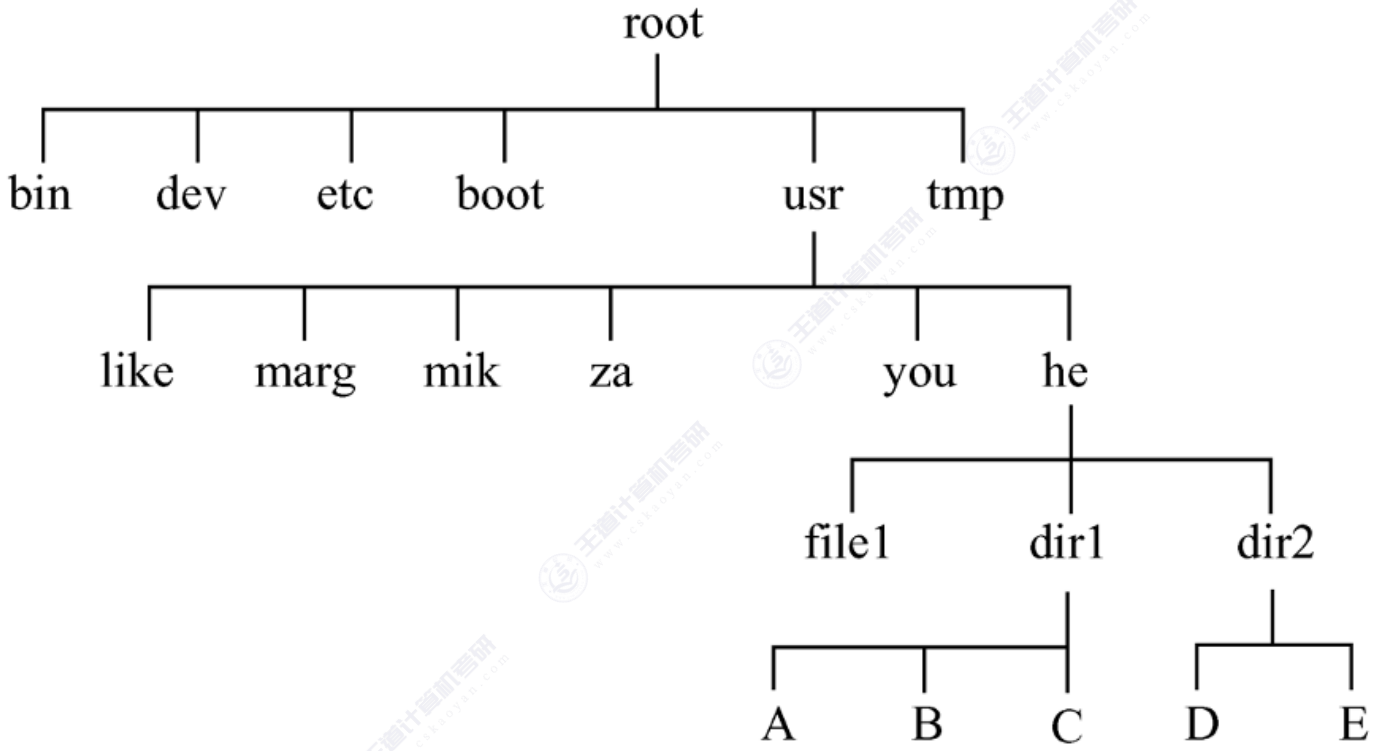
10.某个文件系统中，外存为硬盘。物理块大小为512B，有文件A包含598个记录，每个记录占255B，每个物理块放2个记录。文件A所在的目录如下图所示。

文件目录采用多级树形目录结构，由根目录结点、作为目录文件的中间结点和作为信息文件的树叶组成，每个目录项占127B，每个物理块放4个目录项，根目录的第一块常驻内存。试问：

1）若文件的物理结构采用链式存储方式，链指针地址占2B，那么要将文件A读入内存，至少需要存取几次硬盘？

2）若文件为连续文件，那么要读文件A的第487个记录至少要存取几次硬盘？

3）一般为减少读盘次数，可采取什么措施，此时可减少几次存取操作？



> 【参考答案】：见王道书 4.2_大题6。建议大家看看习题讲解的视频。
>
> 文件管理部分，题目条件多变、灵活。而王道书 4.1、4.2 的课后大题覆盖较为全面，建议大家尽量全做。

11.有一个文件系统如下图1所示。图中的方框表示目录，圆圈表示普通文件。根目录常驻内存，目录文件组织成链接文件，不设FCB，普通文件组织成索引文件。目录表指示下一级文件名及其磁盘地址（各占2B，共4B）。若下级文件是目录文件，指示其第一个磁盘块地址。若下级文件是普通文件，指示其FCB的磁盘地址。每个目录的文件磁盘块的最后4B供拉链使用。下级文件在上级目录文件中的次序在图中为从左至右。每个磁盘块有512B，与普通文件的一页等长。

普通文件的FCB组织如下图2所示。其中，每个磁盘地址占2B，前10个地址直接指示该文件前10页的地址。第11个地址指示一级索引表地址，一级索引表中每个磁盘地址指示一个文件页地址；第12个地址指示二级索引表地址，二级索引表中每个地址指示一个一级索引表地址；第13个地址指示三级索引表地址，三级索引表中每个地址指示一个二级索引表地址。请问：

1）一个普通文件最多可有多少个文件页?

2）若要读文件J中的某一页，最多启动磁盘多少次?

3）若要读文件W中的某一页，最少启动磁盘多少次?

4）根据3），为最大限度减少启动磁盘的次数，可采用什么方法？此时，磁盘最多启动多少次?

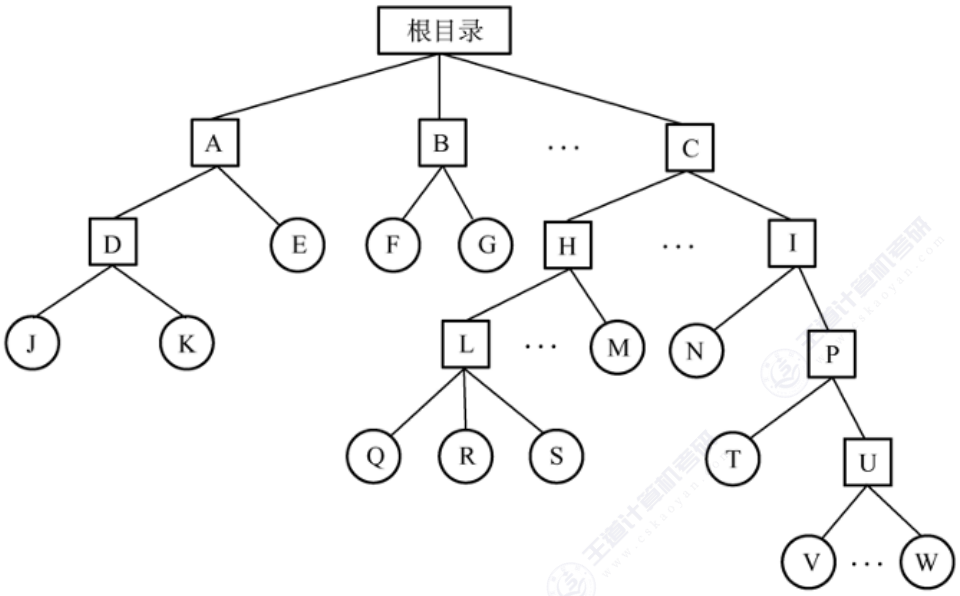


图 A　某树形结构文件系统框图

	该文件的有关描述信息
1	磁盘地址
2	磁盘地址
3	磁盘地址
⋮	…
11	磁盘地址
12	磁盘地址
13	磁盘地址

图 B　FCB 组织

【参考答案】见王道书 4.2_大题5。建议大家看看习题讲解的视频。

文件管理部分，题目条件多变、灵活。而王道书 4.1、4.2 的课后大题覆盖较为全面，建议大家尽量全做。