一堆指令的执行

大纲

Ch4 指令系统

- （一）指令系统的基本概念
- （二）指令格式
  - 1. 指令的基本格式
    - 操作码 —— 指明了动作
    - 操作数（可能有0~n个）
      - 寻址方式 —— 怎么找这个操作数?
      - 形式地址
        - 形式地址（需转成有效地址EA）
        - 指令运行时需转成有效地址EA
    - 根据指令长度是否固定不变，分为两类
      - 定长指令字 —— 每次PC+“1”，“1”固定
      - 变长指令字 —— 每次PC+“1”，“1”不固定
  - 2. 定长操作码指令格式 —— 操作码的位数固定不变
  - 3. 扩展操作码指令格式 —— 操作码的位数可能会改变
- （三）寻址方式
  - 1. 有效地址的概念 —— EA：最终要访问的地址
  - 2. 要寻找什么?
    - 数据寻址：找数据 —— 找到你要操作的数据
    - 指令寻址：找下一条要执行的指令 —— 取决于PC值
  - 3. 常见的寻址方式

本节回顾

寻址方式	有效地址	访存次数(指令执行期间)
隐含寻址	程序指定	0
立即寻址	A即是操作数	0
直接寻址	EA=A	1
一次间接寻址	EA=(A)	2
寄存器寻址	EA=R_i	0
寄存器间接一次寻址	EA=(R)	1
偏移寻址		
转移指令 相对寻址	EA=(PC)+A	1
多道程序 基址寻址	EA=(BR)+A	1
循环程序 变址寻址 数组问题	EA=(IX)+A	1
堆栈寻址	入栈出栈时EA的确定方式不同	硬堆栈不访存，软堆栈访存1次

王道考研/CSKAOYAN.COM

- （四）数据的对齐和大/小端存放方式
  - 数据的对齐
    - 按边界对齐
    - 不按边界对齐
  - 存放方式
    - 大端存储（更符合人类视角）
    - 小端存储
- （五）CISC 和 RISC 的基本概念
  - 从机器级代码中可以看到的显著区别
    - 指令字是否定长？ —— RISC的指令定长，CISC的指令不定长
    - 除了 Load、Store 之外，是否还有其他可以指令可以访存? —— RISC仅有Load、Store 类指令，才可以访问主存
- （六）高级语言程序与机器级代码之间的对应
  - 1. 编译器、汇编器、链接器的基本概念
  - 2. 选择结构语句的机器级表示
  - 3. 循环结构语句的机器级表示
  - 4. 过程（函数）调用对应的机器级表示

解题方法

- 先观察汇编语言，是x86还是MIPS?
  - ①观察是否有注释 —— 通常来说，真题中x86汇编语言不会给太多注释（考研中默认大家懂 x86）
  - ②观察指令长度是否固定
    - x86属于CISC，指令长度不固定
    - MIPS属于RISC，指令长度固定
  - ③观察寄存器名
    - x86的寄存器名为 eax、ebx、ecx、edx...
    - MIPS 的寄存器名为 R[0]、R[1]、R[2]...
- x86
  - 先搞懂C语言
  - 基于C语言的逻辑分析机器指令
    - 有没有分支结构? —— 观察有没有 jxxx 指令
    - 有没有循环结构?
      - 观察有没有 jxxx 指令
      - 观察有没有 loopxxx 指令
    - 有没有函数调用?
      - 观察有没有 call 指令
      - 观察有没有 ret 指令
    - 需不需要访问函数调用参数？观察在汇编语言中如何访问调用参数? —— 注：函数调用参数一般在 [ebp+8]、[ebp+12] 等位置
    - 需不需要定义局部变量？观察在汇编语言中如何访问局部变量? —— 注：局部变量的存储地址一般在 [ebp-4]、[ebp-8]、[ebp-12] 等位置
- MIPS
  - 先搞懂C语言
  - 基于C语言的逻辑分析机器指令
    - 有没有分支结构? —— 观察有没有 bxxx 指令 —— 注：MIPS汇编语言中，转移类指令是以 b 开头的，表示“branch”，即 分支。指令原理与x86的 jxxx 相同
    - 有没有循环结构? —— 观察有没有 bxxx 指令
    - 有没有函数调用? —— 注：MIPS汇编语言指令通常不考函数调用，重点关注 x86 的函数调用
  - 考不考流水线?

可参考强化录播内容（指令流水线）

- 对指令进行分类 —— 五类指令，每一类指令的运行原理要搞清楚
- 安排指令流水线
- 分析数据冒险、控制冒险

基础知识

- x86汇编指令（重点关注 intel 格式）
  - 算数、逻辑运算类指令 —— 加、减、乘、除、左移、右移 等指令
  - 分支结构（if/else）
    - 无条件转移（goto） —— jmp （无条件转移指令），类似于C语言的goto
    - 条件转移指令（if/else）
      - ① cmp A，B （本质是 A-B，A-B 的标志位信息会存放到PSW中）
      - ② jxxx （转移指令指令）
        - 无符号数减法关注 CF、ZF
          - A==B —— 需满足 ZF==1
          - A!=B —— 需满足 ZF==0
          - A>=B —— 需满足 CF==0
          - A<=B —— 需满足 CF==1 || ZF==1
          - A>B —— 需满足 CF==0 && ZF==0
          - A<B —— 需满足 CF==1
        - 有符号数减法关注 OF、SF、ZF
          - A==B —— 需满足 ZF==1
          - A!=B —— 需满足 ZF==0
          - A>=B —— 需满足SF==OF
          - A<=B —— 需满足SF!=OF || ZF==1
          - A>B —— 需满足SF==OF && ZF==0
          - A<B —— 需满足SF!=OF && ZF==0

转移类指令，通常采用相对寻址。用补码表示偏移量。补码的值通常意味着PC要 往前/往后 跳多少个地址（注意：偏移量的单位可能是字节，也可能是指令字长）

x86属于CISC，指令字长不固定，因此转移类指令中一定是以“字节”为单位描述PC的偏移量

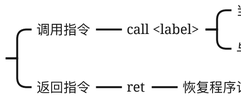
对比

  - 循环结构（for、while）
    - 条件转移指令 jxxx 可以用于实现循环 —— 注：在 x86 汇编语言中，循环结构通常是用jxxx指令实现的，很少使用 loopxxx 指令
    - loopx 指令 —— 原理同转移指令
  - 函数调用/返回
    - 调用指令 —— call <label>
      - 当前PC值压栈保存，再修改PC的值，跳转到<label>
      - 与无条件转移指令jmp的区别 —— call 指令会将函数返回地址(当前的PC)入栈
    - 返回指令 —— ret —— 恢复程序计数器PC，返回原函数
- MIPS 汇编指令
  - 常见指令
    - ①算数、逻辑运算类指令
      - 加 —— OP：add
      - 减 —— OP：sub
      - 乘 —— OP：mul
      - 除 —— OP：div
      - 左移 —— OP：shl
      - 右移 —— OP：shr
    - ②转移类指令 —— bxxx —— b开头，意思是 branch

转移类指令，通常采用相对寻址。用补码表示偏移量。补码的值通常意味着PC要 往前/往后 跳多少个地址（注意：偏移量的单位可能是字节，也可能是指令字长）

MIPS属于RISC，指令字长固定，因此转移类指令中可能会以“指令字长”为单位描述PC的偏移量（参考 2013年真题）

    - ③Load类指令 —— lxxx
      - L 开头，意思是 load
      - 根据读取长度不同，“xxx”部分可能会变，例如从内存读取一个字可以使用指令 lw，这里的 w 指“word”
    - ④Store类指令 —— sxxx
      - s开头，意思是 store
      - 根据存储长度不同，“xxx”部分可能会变，例如往内存写一个字可以使用指令 sw，这里的 w 指“word”
  - 结合五段式指令流水线考察
    - 数据冒险 —— 什么指令可能导致数据冒险?
      - 前面的指令：写某个寄存器 —— WB，第五个段
      - 后面的指令：读（同一个）寄存器 —— ID，第二个段
      - 分析思路 —— 一条一条指令从前往后分析，如果一条指令写了某个寄存器，则观察与之相邻的后3条指令中，有没有哪条指令需要读同一个寄存器
    - 控制冒险 —— 什么指令可能导致控制冒险?
      - 转移类指令
        - M，第四个段改变PC值
        - 处理控制冒险的方法：停三个周期，再取下一条指令（IF段），就不会发生控制冒险
      - 注：call、ret 也属于转移类指令，也会在 M 段修改PC值



橙色：可能导致数据冒险

绿色：可能导致控制冒险

如何分析 数据冒险、控制冒险？

	IF（取指）	ID（译码/取数）	EX（执行/计算地址）	M（访存）	WB（写回）
Load指令	取指令	译码、取数（可能会读寄存器）	计算访存地址	从主存（Data Cache）读数据	将数据写回某个寄存器
Store指令	取指令	译码、取数（可能会读寄存器）	计算访存地址	往主存（Data Cache）写数据	空
转移类指令	取指令	译码、取数（可能会读寄存器）	计算转移地址	更新PC值	空
运算类指令	取指令	译码、取数（可能会读寄存器）	执行（算数运算、逻辑运算等）	空	将数据写回某个寄存器

注：熟悉四类指令在各个段分别做了什么

王道考研/CSKAOYAN.COM

x86汇编语言，AT&T 和 intel 格式的区别（截至目前，历年真题中都是intel格式）

	AT&T 格式	Intel 格式
目的操作数d、源操作数s	op s, d 注：源操作数在左，目的操作数在右	op d, s 注：源操作数在右，目的操作数在左
寄存器的表示	mov %ebx, %eax 注：寄存器名之前必须加"%"	mov eax, ebx 注：直接写寄存器名即可
立即数的表示	mov $985, %eax 注：立即数之前必须加“$”	mov eax, 985 注：直接写数字即可
主存地址的表示	mov %eax , (af996h) 注：用 “小括号”	mov [af996h], eax 注：用 “中括号”

读写长度的表示	movb $5, (af996h)
	movw $5, (af996h)
	movl $5, (af996h)
	addb $4, (af996h)
	注：指令后加 b、w、l 分别表示读写长度为byte、word、dword
		mov byte ptr [af996h], 5
		mov word ptr [af996h], 5
		mov dword ptr [af996h], 5
		add byte ptr [af996h], 4
		注：在主存地址前说明读写长度byte、word、dword

主存地址偏移量的表示	movl -8(%ebx), %eax *注：偏移量(基址)* movl 4(%ebx, %ecx, 32), %eax *注：偏移量(基址,变址,比例因子)*	mov eax, [ebx - 8] *注：[基址+偏移量]* mov eax, [ebx + ecx*32 + 4] *注：[基址+变址*比例因子+偏移量]*

王道考研——计算机组成原理

WWW.CSKAOYAN.COM

强化P4：一堆指令的执行

2009	2010	2011	2012	2013	2014	2015	2016
中断控制方式的处理过程；DMA控制方式的处理过程。	指令格式； 寻址范围； 指令执行的微操作过程	C语言中常见变量的表示；强制类型转换； 补码加法的应用； 溢出判断。	CPU性能指标的计算； 引入Cache后的访存原理； 虚拟内存的工作原理； DMA控制方式的工作原理； 低位交叉存储的流水线	CPU性能指标、总线性能指标的计算； 低位交叉存储方式； 总线的突发传送过程； 引入Cache后的访存原理；	结合C语言，读懂各条指令的作用；条件转移指令的工作原理；五段式指令流水线	指令执行的数据通路； 数据通路上各种常见的硬件部件及作用； 控制信号连线；	中断控制方式的处理过程；
指令执行的详细过程；说明每一步的微操作、微命令。并安排合理的时序	Cache和主存的映射； C语言汇总二维数组的存储原理； Cache命中率的计算	虚拟存储系统的地址结构； Cache的工作原理； TLB的工作原理；	数据的移位运算；五段式指令流水线；流水线的“冲突”原因；	条件转移指令的工作原理； CPU内部常见的硬件部件（根据处理逻辑推测）	Cache的工作原理； 指令的溢出判断； 虚拟存储，缺页异常的产生原因； TLB的工作原理；	指令格式； 各步微操作对应的微命令； 微操作的时序安排；	TLB的工作原理； Cache的工作原理； 虚拟存储，缺页异常； Cache淘汰策略、页面淘汰处理；

2017	2018	2019	2020	2021	2022	2023	2024
C语言强制类型转换； 各种数的精度问题、溢出问题	程序定时查询方式的工作过程； 中断查询方式的工作过程； DMA方式的工作过程；	C语言对应的指令序列； 条件转移指令、无条件转移指令、函数调用call 指令的原理；数据的精度、溢出问题	数据的运算：二进制乘法； 溢出问题	指令格式；数据的运算、溢出问题；	一条指令的执行过程；指令执行的电路数据通路原理	Cache、虚拟页式存储	一条指令的执行过程；指令执行的电路数据通路原理
C语言对应的指令序列；比较指令cmp、条件转移指令的工作原理；数据的的运算，算数左移	虚拟存储，地址结构； TLB的工作原理； Cache的工作原理； 有TLB、Cache的地址变换过程	虚拟分页存储； Cache的工作原理；	Cache的工作原理； 结合C语言分析Cache命中情况；	虚拟存储，地址结构； TLB的工作原理；	磁盘+IO控制方式	结合C语言，分析指令序列的工作原理	结合C语言，分析指令序列的工作原理

- 图示说明：

主考第二章	数据的运算；强制类型转换；精度、溢出问题	3+n
主考第三章	Cache、TLB、虚拟分页	9
主考第五章	一条指令的执行过程	7
主考第四章	指令序列的工作过程	6
主考第七章	三种IO控制方式	4
	杂七杂八	3

分 界 线

x86 一堆指令的执行过程

2023年真题

43.（14 分）已知计算机 M 字长为 32 位，按字节编址，采用请求调页策略的虚拟存储管理方式，虚拟地址为 32 位，页大小为 4 KB；数据 Cache 采用 4 路组相联映射方式，数据区大小为 8KB，主存块大小为 32B。现有 C 语言程序段如下：

```
int a[24][64];
……
for ( i = 0; i < 24; i++ )
    for (j = 0; j < 64; j++) a[i][j] = 10;
```

已知二维数组 a 按行优先存放，在虚拟地址空间中分配的起始地址为 0042 2000H，sizeof(int) = 4，假定在 M 上执行上述程序段之前数组 a 不在主存，且在该程序段执行过程中不会发生页面置换。请回答下列问题。

1）数组 a 分布在几个页面中？对于数组 a 的访问，会发生几次缺页异常？页故障地址各是什么？

2）不考虑变量 i 和 j，该程序段的数据访问是否具有时间局部性？为什么？

3）计算机 M 的虚拟地址（A31～A0）中哪几位用作块内地址？哪几位用作 Cache 组号？a[1][0]的虚拟地址是多少？其所在主存块对应的 Cache 组号是多少？

4）数组 a 占用多少主存块？假设上述程序段执行过程中数组 a 的访问不会和其他数据发生

前置题目

2023年真题

Cache 访问冲突，则数组 a 的 Cache 命中率是多少？若将循环中 i 和 j 的次序按如下方式调换：

```
for ( j = 0; j < 64; j++ )
    for ( i = 0; i < 24; i++ ) a[i][j] = 10;
```

则数组 a 的 Cache 命中率又是多少？

44.（9 分）题 43 中的 C 程序段在计算机 M 上的部分机器级代码如下，每个机器级代码行中依次包含指令序号、虚拟地址、机器指令和汇编指令。

```
    for ( i = 0; i < 24; i++ )
 1  00401072    C7 45 F8 00 00 00 00        mov [ebp-8], 0
 2  00401079    EB 09                       jmp 00401084h
 3  0040107B    8B 55 F8                    mov eax, [ebp-8]
    ……          ……                          ……
 7  00401088    7D 32                       jge   004010bch
        for ( j = 0; j < 64; j++ )
 8  0040108A    C7 45 FC 00 00 00 00        mov [ebp-4], 0
    ……          ……                          ……
            a[i][j] = 10;
    ……          ……                          ……
19  004010AE    C7 84 82 00 20 42 00 0A 00 00 00   mov [ecx+edx*4+00422000h], 0Ah
20  ……          ……                          ……
```

请回答下列问题。

2023年真题

1）第 20 条指令的虚拟地址是多少？

2）已知第 2 条 jmp 和第 7 条 jge 都是跳转指令，其操作码分别是 EBH 和 7DH，跳转目标地址分别为 0040 1084H、0040 10BCH，这两条指令都采用什么寻址方式？给出第 2 条指令 jmp 的跳转目标地址计算过程。

3）已知第 19 条 mov 指令的功能为“a[i][j]←10”，其中 ecx 和 edx 为寄存器名，0042 2000H 是数组 a 的首地址，指令中源操作数采用什么寻址方式？已知 edx 中存放的是变量 j，ecx 中存放的是什么？根据该指令的机器码判断计算机 M 采用的是大端还是小端方式。

4）第一次执行第 19 条指令时，取指令过程中是否会发生缺页异常？为什么？

2019年真题

45.（16 分）已知 $f(n)=n!=n\times(n-1)\times(n-2)\times\cdots\times2\times1$，计算 $f(n)$的 C 语言函数 f1 的源程序（阴影部分）及其在 32 位计算机 M 上的部分机器级代码如下：

```
int   f1(int n){
1     00401000    55              push  ebp
      ...         ...             ...
      if(n>1)
11    00401018    83 7D 08 01     cmp  dword ptr [ebp+8],1
12    0040101C    7E 17           jle  f1+35h (00401035)
      return n*f1(n-1);
13    0040101E    8B 45 08        mov  eax, dword ptr [ebp+8]
14    00401021    83 E8 01        sub  eax, 1
15    00401024    50              push  eax
16    00401025    E8 D6 FF FF FF  call  f1 ( 00401000)
      ...         ...             ...
19    00401030    0F AF C1        imul  eax, ecx
20    00401033    EB 05           jmp  f1+3Ah (0040103a)
      else return 1;
21    00401035    B8 01 00 00 00  mov  eax,1
      }
      ...         ...             ...
26    00401040    3B EC           cmp  ebp, esp
      ...         ...             ...
30    0040104A    C3              ret
```

其中，机器级代码行包括行号、虚拟地址、机器指令和汇编指令，计算机 M 按字节编址，int 型数据占 32 位。请回答下列问题：

（1）计算 $f(10)$需要调用函数 f1 多少次？执行哪条指令会递归调用 f1？

（2）上述代码中，哪条指令是条件转移指令？哪几条指令一定会使程序跳转执行？

（3）根据第 16 行的 call 指令，第 17 行指令的虚拟地址应是多少？已知第 16 行的 call 指令采用相对寻址方式，该指令中的偏移量应是多少（给出计算过程）？已知第 16 行的 call 指令的后 4 字节为偏移量，M 是采用大端方式还是采用小端方式？

（4）$f(13)=6227020800$，但 f1(13)的返回值为 1932053504，为什么两者不相等？要使 f1(13)能返回正确的结果，应如何修改 f1 的源程序？

（5）第 19 行的 imul 指令（带符号整数乘）的功能是 R[eax]←R[eax]×R[ecx]，当乘法器输出的高、低 32 位乘积之间满足什么条件时，溢出标志 OF = 1？要使 CPU 在发生溢出时转异常处理，编译器应在 imul 指令后应加一条什么指令？

2017年真题

44.（10 分）在按字节编址的计算机 M 上，题 43 中 f1 的部分源程序（阴影部分）与对应的机器级代码（包括指令的虚拟地址）如下图所示。

```
            int f1( unsigned n)
1           00401020    55          push ebp
            …           …           …
                for(unsigned i=0; i<= n -1; i++)
            …           …           …
20          0040105E    39 4D F4    cmp dword ptr [ebp-0Ch],ecx
            …           …           …
                    power *= 2;
            …           …           …
23          00401066    D1 E2       shl  edx,1
            …           …           …
                return sum;
            …           …           …
35          0040107F    C3          ret
```

其中，机器级代码行包括行号、虚拟地址、机器指令和汇编指令。请回答下列问题。

（1）计算机 M 是 RISC 还是 CISC？为什么？

（2）f1 的机器指令代码共占多少字节？要求给出计算过程。

（3）第 20 条指令 cmp 通过 i 减 n-1 实现对 i 和 n-1 的比较。执行 f1(0)过程中，当 i = 0 时，cmp 指令执行后，进/借位标志 CF 的内容是什么？要求给出计算过程。

（4）第 23 条指令 shl 通过左移操作实现了 power*2 运算，在 f2 中能否也用 shl 指令实现 power*2？为什么？

分 界 线

MIPS/RISC-V 一堆指令的执行（常结合指令流水线考察）

2014年真题

44.（12 分）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？

2012年真题

44．某 16 位计算机中，带符号整数用补码表示，数据 Cache 和指令 Cache 分离。题 44 表给出了指令系统中部分指令格式，其中 Rs 和 Rd 表示寄存器，mem 表示存储单元地址，(x)表示寄存器 x 或存储单元 x 的内容。

指令系统中部分指令格式

名称	指令的汇编格式	指令功能
加法指令	ADD　Rs，Rd	(Rs) + (Rd)->Rd
算术/逻辑左移	SHL　Rd	2*(Rd)->Rd
算术右移	SHR　Rd	(Rd)/2->Rd
取数指令	LOAD　Rd, mem	(mem)->Rd
存数指令	STORE　Rs, mem	(Rs)->mem

该计算机采用 5 段流水方式执行指令，各流水段分别是取指(IF)、译码/读寄存器(ID)、执行/计算有效地址（EX)、访问存储器（M）和结果写回寄存器（WB），流水线采用“按序发射，按序完成”方式，没有采用转发技术处理数据相关，并且同一个寄存器的读和写操作不能在同一个时钟周期内进行。请回答下列问题：

2012年真题

1）若 int 型变量 x 的值为-513，存放在寄存器 R1 中，则执行指令“SHR R1”后，R1 的内容是多少？（用十六进制表示）

2）若某个时间段中，有连续的 4 条指令进入流水线，在其执行过程中没有发生任何阻塞，则执行这 4 条指令所需的时钟周期数为多少？

3）若高级语言程序中某赋值语句为 x = a + b，x、a 和 b 均为 int 型变量，它们的存储单元地址分别表示为[x]、[a]和[b]。该语句对应的指令序列及其在指令流水线中的执行过程如下图所示。

I_1	LOAD	R1, [a]
I_2	LOAD	R2, [b]
I_3	ADD	R1, R2
I_4	STORE	R2, [x]

	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

则这 4 条指令执行过程中，I_3 的 ID 段和 I_4 的 IF 段被阻塞的原因各是什么？

4）若高级语言程序中某赋值语句为 x = x*2 + a，x 和 a 均为 unsigned int 类型变量，它们的存储单元地址分别表示为[x]、[a]，则执行这条语句至少需要多少个时钟周期？要求模仿题 44 图画出这条语句对应的指令序列及其在流水线中的执行过程示意图。