P3_数据的表示和运算大题

大纲：数据的表示和运算

（一）数制与编码

- 1.进位计数制及其相互转换
  - 二进制
  - 八进制
  - 十进制
  - 十六进制
  - 相互转换
    - 二、八、十六 之间的转换很简单
    - 二转十 — 根据每位权值展开即可
    - 十 转 二
      - 好拆凑的 — 拆凑法（熟悉每个二进制位的权重）
      - 不好拆凑的
        - 整数部分：除2留余数，先留是低位后留是高位
        - 小数部分：乘2取整，先取是高位后取是低位
- 2.定点数的编码表示
  - 无符号数的表示 — 重点关注无符号数的“减法”如何实现，以及无符号数加法/减法的溢出判断
  - 带符号整数的表示
    - 原
    - 反
    - 补 — 重点关注补码的“减法”如何实现，以及补码加法/减法的溢出判断
      - 把符号位和数值位一起编码的方法
    - 移 — 注意偏移量是多少
  - 定点小数 — 仅用于浮点数尾数，关注加减法即可

（二）运算方法和运算电路

- 1. 基本运算部件
  - 加法器
  - 算数逻辑部件（ALU）
- 2. 加/减运算
  - 补码加/减运算器
  - 标志位的生成 — CF、SF、ZF、OF
- 3. 乘/除运算
  - 乘/除法运算的基本原理
  - 乘法电路和除法电路的基本结构

（三）数据的表示和运算

- 1. 无符号整数的表示和运算
  - 表示
  - 运算 — 加、减、乘、除
  - 溢出判断
- 2. 带符号整数的表示和运算
  - 表示 — 原、反、补、移
  - 运算（仅关注补码即可）
    - 加、减、乘、除
    - 溢出判断

（四）浮点数的表示和运算

- 1.浮点数的表示：IEEE754 标准 — 结构：符号位+阶码+尾数
  - float：1+8+23
  - double：1+11+52
  - long double：1+15+64
- 2.浮点数的加/减运算
  - 对阶：小阶向大阶对齐
  - 尾数加减：原码定点数的加减法
  - 规格化：尾数规格化为 1.xxxxx 的形式
    - 尾数每左规一位，阶码 - 1
    - 尾数每右规一位，阶码 + 1
  - 舍入：尾数右规时，可能导致精度丢失，需考虑舍入问题
  - 判溢出
    - 尾数右规时，阶码超出可表示的最大值，上溢 — 发生溢出异常
    - 尾数左规时，阶码低于可表示的最小值，下溢 — 当做机器0处理

常见考法

- 结合C语言程序，分析“数据的表示和运算”
  - 【2020】43题
  - 【2017】43题
  - 【2011】43题
  - 常规考法
- 结合机器级代码（通常也会给出对应的C语言程序），分析“数据的表示和运算”在指令运行过程中的作用
  - 【2021】43题
  - 【2019】45题
  - 【2017】44题
  - 【2014】44题
  - 【2013】44题
  - 加减运算在“条件转移类指令”中的运用

大题重点关注

数据的表示

- 无符号整数
  - unsigned short
  - unsigned int
- 带符号整数
  - short
  - int
- C语言常见整数类型 — 记住16位表示范围
  - 记住
    - 2^16 = 65536
    - 2^15=32768
  - 带符号 short — -32768~32767
  - 无符号 unsigned short — 0~65535

- IEEE754浮点数 — 结构：符号位+阶码+尾数
  - float：1+8+23
  - double：1+11+52
  - long double：1+15+64
- C语言数据类型转换
  - 整数之间的转换
    - 长度相同 — 机器数不变，解释方式改变（带符号整数、无符号整数）
    - 短变长 — 先扩展（无符号整数零扩展、带符号整数符号扩展），再解释
    - 长变短 — 先截断（留低位），再解释
  - 整数和浮点数之间的转换
    - 整数转浮点数 — 写出人类视角的科学计数法 1.xxxx，再转为浮点数，截断尾部时采用 0 舍 1 入原则
    - 注意：精度丢失问题
    - 浮点数转整数 — 写出二进制小数，去掉小数部分，整数部分保留更低的位数
    - 注意：溢出问题，精度丢失（小数点后的部分）

数据的运算

- 加减法
  - 无符号整数
    - 加法规则
      - 机算 — 按位相加
      - 手算 — 带入十进制计算
    - 减法规则
      - 机算 — 减数按位取反末位加1，减法变加法
      - 手算 — 带入十进制计算
    - 溢出问题
      - 手算 — 判断加减法的结果是否超出无符号数合法范围
      - 机算 — 根据CF标志位判断，CF=最高位产生的进位 异或 sub
  - 带符号整数
    - 加法规则
      - 机算 — 按位相加
      - 手算 — 带入十进制计算
    - 减法规则
      - 机算 — 减数按位取反末位加1，减法变加法
      - 手算 — 带入十进制计算
    - 溢出问题
      - 手算 — 判断加减法的结果是否超出带符号数合法范围
      - 机算 — 根据OF标志位判断，OF=最高位产生的进位 异或 次高位产生的进位
  - IEEE754浮点数 — 小题考察很多（可通过王道书课后小题复习）
- 乘法
  - 无符号整数
    - 机算 — 注意：可用逻辑左移替代 *2
    - 溢出判断？
      - n 位乘 n 位，若用 2n 位保存乘积，则不会溢出
      - n 位乘 n 位，用 2n 位保存中间结果，最后截取末尾 n 位作为最终的乘积，可能会溢出
        - 手算判溢出 — 带入十进制计算乘法结果，判断该结果是否超出了 n 位“无符号数”所能表示的范围，若超出，则溢出
        - 机器判溢出 — 两个无符号数乘法：n 位乘 n 位，用 2n 位保存中间结果，仅当前n位都是0时，不溢出
  - 带符号整数
    - 机算 — 注意：可用算数左移替代 *2
    - 溢出判断？
      - n 位乘 n 位，若用 2n 位保存乘积，则不会溢出
      - n 位乘 n 位，用 2n 位保存中间结果，最后截取末尾 n 位作为最终的乘积，可能会溢出
        - 手算判溢出 — 带入十进制计算乘法结果，判断该结果是否超出了 n 位“带符号数”所能表示的范围，若超出，则溢出
        - 机器判溢出 — 两个有符号补码乘法：n 位乘 n 位，用 2n 位保存中间结果，仅当前 n+1 位全1或全0时，不溢出
- 除法
  - 无符号整数 — 机算 — 注意：可用逻辑右移替代 /2
  - 带符号整数 — 机算 — 注意：可用算数右移替代 /2

常问的难点

1. 无符号数的加法、减法怎么算？
   - 在硬件看来，对无符号数的加减法，对有符号补码的加减法都是一样的计算方法
   - 加法：直接 N bit二进制按位相加
   - 减法：计算 “A - B”，可转化为等价的加法——将 B 的 N bit 全部按位取反末位+1，然后与A相加
2. 遇到乘法怎么算？
   - 通常可带入十进制手算
   - 408大纲不要求掌握浮点数乘法
3. 遇到除法怎么算？
   - 通常可带入十进制手算
   - 408大纲不要求掌握浮点数除法
4. 溢出怎么判断？
   - 加法/减法
     - 手算判溢出 — 带入十进制计算结果，判断该结果是否超出了 n 位所能表示的范围，若超出，则溢出
     - 机器判溢出
       - 无符号数加/减：运算后若 CF=1，则溢出；若 CF=0，则不溢出
       - 有符号数补码加/减：运算后若 OF=1，则溢出；若OF=0，则不溢出
   - 乘法
     - n 位乘 n 位，若用 2n 位保存乘积，则不会溢出
     - n 位乘 n 位，用 2n 位保存中间结果，最后截取末尾 n 位作为最终的乘积，可能会溢出
       - 手算判溢出 — 带入十进制计算乘法结果，判断该结果是否超出了 n 位所能表示的范围，若超出，则溢出
       - 机器判溢出
         - 两个无符号数乘法：n 位乘 n 位，用 2n 位保存中间结果，仅当前n位都是0时，才不溢出。
         - 两个有符号补码乘法：n 位乘 n 位，用 2n 位保存中间结果，仅当前 n+1 位全1或全0时，不溢出
   - 除法 — 可能溢出，但是，整数的除法不可能溢出，只有小数的除法可能溢出，而408不要求掌握浮点数除法，因此不需太关注除法的溢出问题
5. 精度丢失怎么判断？
   - 浮点数的精度：看尾数的位数
     - 单精度浮点型 float：尾数位数为 23+1 bit，因此数值部分可以表示 24bit 的精度
     - 双精度浮点型 double：尾数位数为 52+1 bit，因此数值部分可表示 53bit 的精度
     - 临时浮点型 long double：尾数位数为 64+1 bit，因此数值部分可表示 65bit 的精度
     - 注：IEEE 754标准中，尾数是规格化的，隐含了一个最高位的 1
   - 定点整数的精度：看数值位的位数
     - 16位short型：数值部分的精度为 15 bit
     - 32位int型：数值部分的精度为 31 bit
     - 32位 unsigned int 型：数值部分的精度为 32bit
     - 64位long型：数值部分的精度为 63 bit
   - 整数转浮点数：数值部分精度更高的类型，转为数值部分精度更低的类型，既有可能发生精度丢失
   - 浮点数转整数：如果有浮点数有小数部分，则可能丢失精度
6. CF、ZF、OF、SF 傻傻分不清楚？
   - CF
     - 含义：进位/借位标志，表示无符号数的加减法是否发生了进位或借位，当CF=1时，说明无符号数的加减运算发生了进位或借位，也即发生了溢出
     - 硬件的计算方法：CF = 最高位产生的进位 ⊕ sub
       - sub=1表示减法
       - sub=0表示加法
     - 注意：CF位对有符号数的加减法无意义
   - ZF
     - 含义：表示运算结果是否为0，ZF=1表示运算结果为0，ZF=0表示运算结果非0
     - 硬件的计算方法：两个数的运算结果为n bit，只有n bit全为0时，ZF =1
   - OF
     - 含义：有符号数的加减运算是否发生了溢出，OF=1时，说明发生了溢出
     - 硬件的计算方法：OF= 最高位产生的进位 ⊕ 次高位产生的进位
     - 注意：OF位对无符号数的加减法无意义
   - SF
     - 含义：有符号数加减运算结果的正负性，SF=0表示运算结果为正数，SF=1表示运算结果为负数
     - 硬件的计算方法：SF = 最高位的本位和
     - 注意：SF位对无符号数的加减法无意义

王道考研——计算机组成原理

WWW.CSKAOYAN.COM

强化P3：数据的表示和运算大题总结

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

映射关系：真题→王道书

【2011】43题——王道书2.2.5_大题4
【2020】43题——王道书2.2.5_大题5
【2017】43题——王道书2.3.5_大题5

【2021】43题——王道书4.2.3_大题9
【2019】45题——王道书4.3.5_大题2
【2017】44题——王道书4.3.5_大题1
【2014】44题——王道书5.6.6_大题4
【2013】44题——王道书4.2.3_大题7

分　界　线

主要考数据运算的大题

2020年真题

43.（13 分）有实现 x×y 的两个 C 语言函数如下：

```
unsigned umul (unsigned x, unsigned y)   { return x*y; }
int imul (int x, int y)  {return x * y; }
```

假定某计算机 M 中 ALU 只能进行加减运算和逻辑运算。请回答下列问题。

（1）若 M 的指令系统中没有乘法指令，但有加法、减法和位移等指令，则在 M 上也能实现上述两个函数中的乘法运算，为什么？

（2）若 M 的指令系统中有乘法指令，则基于 ALU、位移器、寄存器以及相应控制逻辑实现乘法指令时，控制逻辑的作用是什么？

（3）针对以下三种情况：a）没有乘法指令；b）有使用 ALU 和位移器实现的乘法指令；c）有使用阵列乘法器实现的乘法指令，函数 umul()在哪种情况下执行时间最长？哪种情况下执行的时间最短？说明理由

（4）n 位整数乘法指令可保存 $2n$ 位乘积，当仅取低 n 位作为乘积时，其结果可能会发生溢出。当 $n = 32$、$x = 2^{31} - 1$、$y = 2$ 时，带符号整数乘法指令和无符号整数乘法指令得到的 x×y 的 $2n$ 位乘积分别是什么（用十六进制表示）？此时函数 umul()和 imul()的返回结果是否溢出？对于无符号整数乘法运算，当仅取乘积的低 n 位作为乘法结果时，如何用 $2n$ 位乘积进行溢出判断？

2017年真题

43.（13 分）已知 $f(n)=\sum_{i=0}^{n}2^i=2^{n+1}-1=\overbrace{11\cdots1}^{n+1位}B$，计算 $f(n)$ 的 C 语言函数 f1 如下：

```
int f1(unsigned n){
    int sum=1, power=1;
    for(unsigned i=0;i<=n-1;i++){
        power *= 2;
        sum += power;
    }
    return sum;
}
```

将 f1 中的 int 都改为 float，可得到计算 $f(n)$ 的另一个函数 f2。假设 unsigned 和 int 型数据都占 32 位，float 采用 IEEE 754 单精度标准。请回答下列问题。

（1）当 n = 0 时，f1 会出现死循环，为什么？若将 f1 中的变量 i 和 n 都定义为 int 型，则 f1 是否还会出现死循环？为什么？

（2）f1(23)和 f2(23)的返回值是否相等？机器数各是什么（用十六进制数表示）？

（3）f1(24)和 f2(24)的返回值分别为 33 554 431 和 33 554 432.0，为什么不相等？

（4）f(31) = $2^{32}-1$，而 f1(31)的返回值却为-1，为什么？若使 f1(n)的返回值与 f(n)相等，则最大的 n 是多少？

（5）f2(127)的机器数为 7F80 0000H，对应的值是什么？若使 f2(n)的结果不溢出，则最大的 n 是多少？若使 f2(n)的结果精确（无舍入），则最大的 n 是多少？

2011年真题

43.（11 分）假定在一个 8 位字长的计算机中运行如下 C 程序段：

```
unsigned int  x=134;
unsigned int  y=246;
int  m=x;
int  n=y;
unsigned int  z1=x-y;
unsigned int  z2=x+y;
int  k1=m-n;
int  k2=m+n;
```

若编译器编译时将 8 个 8 位寄存器 R1～R8 分别分配给变量 x、y、m、n、z1、z2、k1 和 k2。请回答下列问题。（提示：带符号整数用补码表示。）

（1）执行上述程序段后，寄存器 R1、R5 和 R6 的内容分别是什么（用十六进制表示）？

（2）执行上述程序段后，变量 m 和 k1 的值分别是多少（用十进制表示）？

（3）上述程序段涉及带符号整数加/减、无符号整数加/减运算，这四种运算能否利用同一个加法器辅助电路实现？简述理由。

（4）计算机内部如何判断带符号整数加/减运算的结果是否发生溢出？上述程序段中，哪些带符号整数运算语句的执行结果会发生溢出？

分　界　线

数据运算&其他考点

2021年真题

43. （15 分）假定计算机 M 字长为 16 位，按字节编址，连接 CPU 和主存的系统总线中地址线为 20 位、数据线为 8 位，采用 16 位定长指令字，指令格式及其说明如下：

格式	6 位	2 位	2 位	2 位	4 位	指令功能或指令类型说明
R 型	000000	rs	rt	rd	op1	R[rd] ← R[rs] op1 R[rt]
I 型	op2	rs	rt	imm		含ALU运算、条件转移和访存操作3类指令
J 型	op3	target				PC 的低 10 位 ← target

其中，op1～op3 为操作码，rs, rt 和 rd 为通用寄存器编号，R[r]表示寄存器 r 的内容，imm 为立即数，target 为转移目标的形式地址。请回答下列问题。

1）ALU 的宽度是多少位？可寻址主存空间大小为多少字节？指令寄存器、主存地址寄存器（MAR）和主存数据寄存器（MDR）分别应有多少位？

2）R 型格式最多可定义多少种操作？I 型和 J 型格式总共最多可定义多少种操作？通用寄存器最多有多少个？

2021年真题

3）假定 op1 为 0010 和 0011 时，分别表示带符号整数减法和带符号整数乘法指令，则指令 01B2H 的功能是什么（参考上述指令功能说明的格式进行描述）？若 1, 2, 3 号通用寄存器当前内容分别为 B052H, 0008H, 0020H，则分别执行指令 01B2H 和 01B3H 后，3 号通用寄存器内容各是什么？各自结果是否溢出？↵

4）若采用 I 型格式的访存指令中 imm（偏移量）为带符号整数，则地址计算时应对 imm 进行零扩展还是符号扩展？↵

5）无条件转移指令可以采用上述哪种指令格式？↵

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？

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？

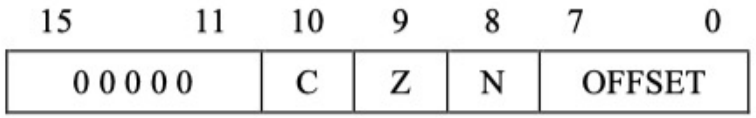
2）已知 sll 指令实现左移功能，数组 A 中每个元素占多少位？

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？

2013年真题

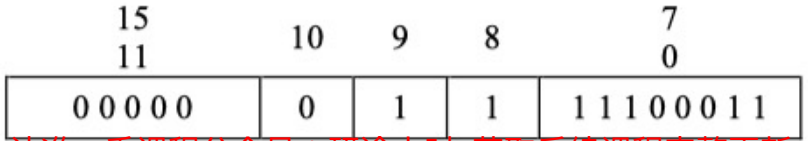
44.（14 分）某计算机采用 16 位定长指令字格式，其 CPU 中有一个标志寄存器，其中包含进位/借位标志 CF、零标志 ZF 和符号标志 NF。假定为该机设计了条件转移指令，其格式如下：



其中，00000 为操作码 OP；C、Z 和 N 分别为 CF、ZF 和 NF 的对应检测位，某检测位为 1 时表示需检测对应标志位，需检测的标志位中只要有一个为 1 就转移，否则不转移，例如，若 C=1，Z=0，N=1，则需检测 CF 和 NF 的值，当 CF=1 或 NF=1 时发生转移；OFFSET 是相对偏移量，用补码表示。转移执行时，转移目标地址为(PC)+2+2×OFFSET；顺序执行时，下条指令地址为(PC)+2。请回答下列问题。

（1）该计算机存储器按字节编址还是按字编址？该条件转移指令向后（反向）最多可跳转多少条指令？

（2）某条件转移指令的地址为 200CH，指令内容如下图所示，若该指令执行时 CF=0，ZF=0，NF=1，则该指令执行后 PC 的值是多少？若该指令执行时 CF=1，ZF=0，NF=0，则该指令执行后 PC 的值又是多少？请给出计算过程。



2013年真题

（3）实现“无符号数比较小于等于时转移”功能的指令中，C、Z 和 N 应各是什么？

（4）以下是该指令对应的数据通路示意图，要求给出图中部件①～③的名称或功能说明。