**2019-2020学年第1学期**

**2018级《计算机组成原理A/B》**

**期末考试试题(A卷)答案**

**考试时间：2019年12月19日**

**一、单项选择题（10分，每小题1分）**

C D B C A D B B A A

**二、判断对错，正确的标记即可，错误的需要给出适当解释。（10分，每小题2分）**

**1. 32位单精度IEEE754格式的浮点数7F800000H（H代表16进制数）表示的是。**

**答：错误。浮点数7F800000H的数符为0、阶码全1为255，而尾数为0，则该数表示正无穷。**

**2.全局指针$gp初值为10008000H，访问空间是1000 8000H到1001 FFFFH之间的内存空间。（H表示十六进制数）**

**答：错误。访问的空间是1000 0000H到1000 FFFFH**

**3. Cache写操作中，采用写直达的方式是将被写数据所在Cache整块放入到写缓冲中，在处理器空闲时将其写入到主存里。**

**答：错误。写直达可以采用写缓冲的方式，将待写数据放到写缓冲中。**

**注意：不是将整个cache块放入写缓冲。**

**4. 单周期指令可以直接采用流水线的方式进行加速。**

**答：错误。单周期指令不可以采用流水线方式。只有多周期指令才可能采用流水线方式。**

**5. MIPS中相对C语言只有很少的决策和循环语句，是因为更多的决策语句意味着更少的代码量，执行更少的操作。**

答：对。详见书P64 - 2.7节小测验1.

**三、综合问答题（30分，每小题5分）**

**1. 简述缩短cpu时间的方法。**

**答案与得分要点：**

**根据公式CPU时间=指令数 \* CPI \* 时钟周期时间。**

**指令数：执行某程序所需的总指令数量。**

**CPI指令数：执行某程序所需的总指令数量每条指令的时钟周期数，表示执行某个程序或者程序片段时每条指令所需的时钟周期平均数。**

**时钟周期：为计算机一个时钟周期的时间，通常是处理器时钟，一般为常数。**

**这个公式清楚地表明，硬件设计者减少一个程序的 CPU 时钟周期数，或减少时钟周期时间，就能改进性能。在后面几章中我们将看到，设计者经常要面对这些因素之间的权衡。许多技术在减少时钟周期数的同时也会引起时钟周期时间的增加。**

**2．简述一个C语言程序转换为可执行程序的4个步骤。**

**答案与得分要点：**

**C语言程序->（编译器）->汇编语言程序->（汇编器）->目标模块：机器语言模块&目标库：例程序->（链接器）->可执行代码：机器语言程序->（加载器）->存储器。**

**3．请解释程序具有的局部性体现在哪些方面。**

**答案与得分要点：**

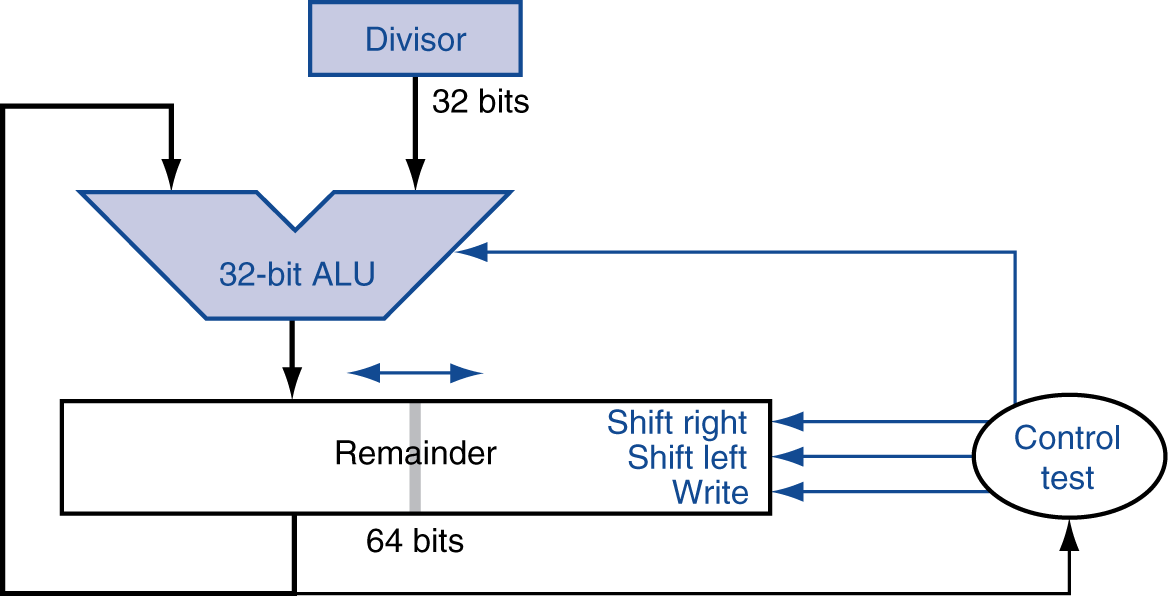
**时间局部性与空间局部性。**

**对两个局部性进行解释。**

**时间局部性：某个数据项在被访问之后可能很快被再次访问的特性**

**空间局部性：某个数据项在被访问之后，与其地址相近的数据项可能很快被访问的特性**

**4．请画出乘除法器原理图，并解释一个除法的执行过程。**

**P127页图3-11**

**余数寄存器左移，上商到余数寄存器的最低位。2分**

**5．MIPS如何得到一个32位的立即数。**

**采用Lui指令，将高16位加载，低16位清零**

**Ori将低16位插入，形成32位立即数。**

**6. 简述控制冒险的三种解决方法。**

**1）假定分支不发生；2）缩短分支延迟；3)动态分支预测。**

**对三种方法适当解释。**

**1)执行分支指令时，假设分支不发生，继续执行顺序的指令流。如果分支发生，就丢弃己经读取并译码的指令，并按分支目标继续执行；如果分支不发生的可能性是 50 %, 同时丢弃的代价很小，那么这种优化方法可以将控制冒险的代价减半。**

**2)为了将分支决策提前，需要提前两个动作：计算分支目标地址和判断分支条件。分支目标地址的计算是比较简单的，我们在 IF/ID 流水线寄存器中已经有了 PC 的值和立即数字段，所以只需要将分支地址计算从 EX 级移到 ID 级就可以了。当然，尽管分支目标地址对所有指令都会计算，但仅在需要时才会使用。**

**判断分支条件比较复杂。为了判断分支的执行条件，需要比较从 ID 级取到的两个寄存器的值是否相等，判断相等的方法可以是先将对应的位进行异或操作，然后将结果按位进行或操。为了把分支条件判断提前到 ID 级，还需要额外的旁路和冒险检测硬件，因为分支条件的判断可能依赖于还在流水线中的结果。**

**3）动态分支预测是通过查找指令的地址观察上一次执行该指令时分支是否发生，如果上次执行时分支发生，就从上次分支发生的地方开始取新的指令。**

**可以采用分支预测缓存分支历史记录表分支预测缓存是小块按照分支指令的地址低位索引的存储器区，其中包括一位或多位数据用以说明最近是否发生过分支。**

**四、综合题（50分）**

**1.（20分）**

**某计算机虚拟地址空间大小为256MB，主存地址空间大小为16MB，页面大小为128KB；Cache采用2路组相联映射方式，共16块；主存与Cache之间交换的块大小为16字（一个字四个字节）。系统运行到某一时刻时，页表的部分内容和Cache的部分内容分别如下图所示，图中物理页号/磁盘地址及标记字段的内容为十六进制形式。请回答下列问题：**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **虚页号** | **有效位** | **物理页/磁盘地址** |  | **组号** | **有效位** | **标记** | **有效位** | **标记** |
| **0H** | **1** | **06H** |  | **0** | **1** | **0200 H** | **0** | **---** |
| **1H** | **1** | **04H** |  | **1** | **0** | **---** | **1** | **0251H** |
| **2H** | **1** | **15 H** |  | **2** | **1** | **04C0 H** | **1** | **032EH** |
| **3H** | **1** | **02 H** |  | **3** | **1** | **01D2 H** | **0** | **---** |
| **4H** | **0** | **---** |  | **4** | **1** | **0640 H** | **1** | **00CDH** |
| **5H** | **1** | **28 H** |  | **5** | **1** | **04DA H** | **1** | **0D7FH** |
| **6H** | **0** | **---** |  | **6** | **0** | **---** | **0** | **---** |
| **7H** | **1** | **32 H** |  | **7** | **1** | **07AB H** | **1** | **0020H** |
| **…** | **…** | **…** |  |  |
| **15H** | **0** | **…** |  |  |
| **16H** | **1** | **23H** |  |  |

**页表的内容 cache的部分内容**

**1）虚拟地址共有几位，那几位表示页号？物理地址共有几位，哪几位表示物理页号？（4分）**

**答：虚拟地址空间大小为256MB（28位），主存地址空间大小为16MB（24位），页面大小为128KB（17位）。**

**故虚拟地址共有28位，高11位表示页号；物理地址共有24位，高7位表示物理页号.**

**2）使用物理地址访问Cache时，给出物理地址的划分格式。（4分）**

**答：Cache采用2路组相联映射方式，共16块，8组（组号3位）；主存与Cache之间交换的块大小为16字（一个字四个字节）（块内地址6位）。**

**故主存地址的格式为：**

|  |  |  |
| --- | --- | --- |
| **主存标记** | **组号索引** | **块内地址** |
| **15** | **3** | **6** |

**3）使用虚拟地址002C050H访问时，能否从Cache中读取到数据？要求给出推导过程。（6分）**

**答：虚拟地址002C050H对应的虚拟页号是：001，查页表有效位为1，表明该虚页已调入主存，对应的主存页号是04H，与页内地址拼接，主存地址是08C050H，二进制为0000 1000 1100 0000 0101 0000，主存标记为0000 1000 1100 000，十六进制为0460H，组号为001，查cache主存映射表，第一组内有数据，但标记不同，没命中，所以从cache中读不到数据。**

**4）假定为该机配置一个全相联的TLB，该TLB共可存放4个页表项，若其采用LRU替换算法，当前内容如下图所示，此时依次访问虚拟地址027BAC0H和0110140H，问接下来继续访问02A0020H所在的页面是否在主存中？要求说明理由。注：H表示16进制数。（6分）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **有效位** | **脏位** | **引用位** | **标记** | **物理页面地址** |
| **1** | **1** | **1** | **6F3H** | **3FH** |
| **1** | **0** | **0** | **025H** | **08H** |
| **1** | **1** | **1** | **09EH** | **1DH** |
| **1** | **1** | **0** | **008H** | **07H** |

**TLB内容**

**答：**

**虚拟地址027BAC0H，对应的页标记为0000 0010 011，十六进制为013H；**

**虚拟地址0110140H，对应的页标记为0000 0001 000，十六进制为008H；**

**虚拟地址02A0020H，对应的页标记为0000 0010 101，十六进制为015H.**

**访问027BAC0H时，TLB没命中，需要把虚页调入内存，并将TLB页标记为025H的页淘汰出内存（脏位和引用位都为0，表明最近未被修改或引用）；访问0110140H 时，TLB中的页标记有008H，命中；当访问02A0020H 时，其对应的页标记为015H，查TLB，没命中，页面不在主存。**

**2.（30分）已知一段MIPS指令序列如下：**

**L2： lw $s3, 4($s2) # (1)**

**add $s5, $s3, $s1 # (2)**

**addi $s7, $s5, 1 # (3)**

**beq $s5, $s6, Exit # (4)**

**sw $s5, 4($s2) # (5)**

**j L2 # (6)**

**Exit: # (7)**

**请回答下列问题：**

1. **上述（1）、（2）、（3）语句的寻址方式。（3分）**

**答案与得分点：（1）基址寻址；（2）寄存器寻址；（3）立即数寻址。**

1. **指出指令（4）和（6）的两点区别。（4分）**

**答案与得分点：**

**（4）是条件转移指令，目标地址是PC+4+位移；（6）**是无条件跳转指令。目标地址为PC高4位：位移：00。

**3）若指令（1）所在的内存地址是0，则指令（6）所在的内存地址是多少。（3分）**

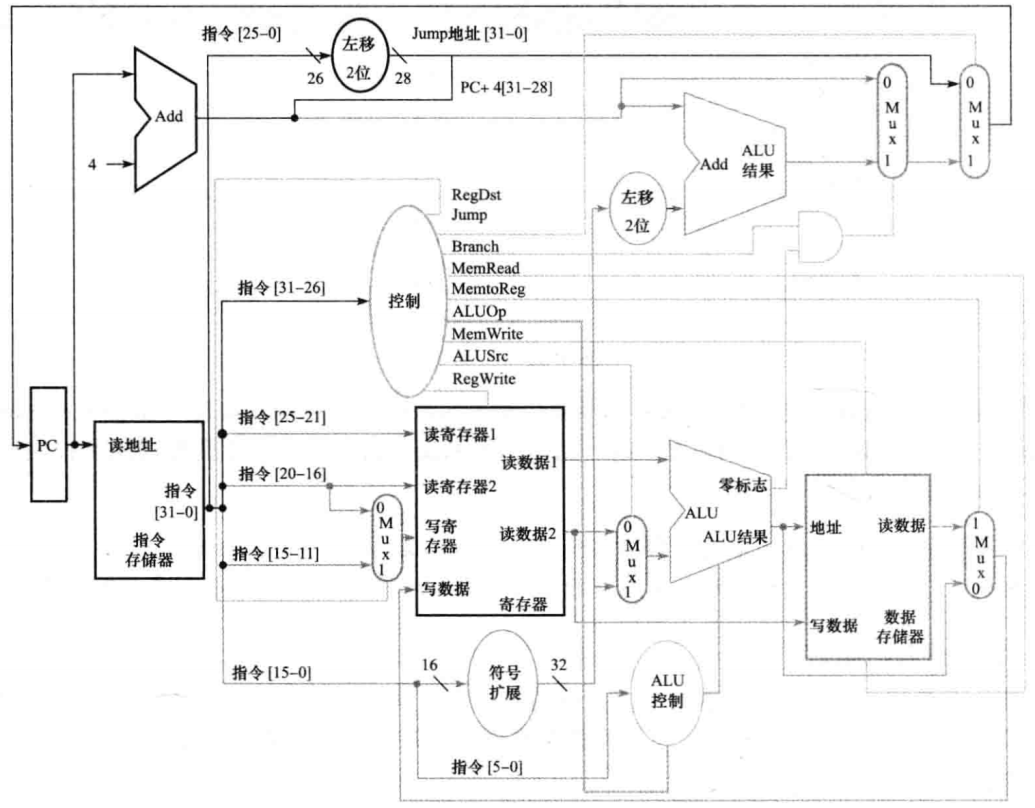
**答案与得分点：**

**20（14H）。（3分）**

**4）请画出一个能实现上述指令的简单的MIPS体系结构数据通路硬件结构图（数据通路包括五个阶段：IF取指令；ID指令译码、读寄存器堆；EX执行或计算地址；MEM数据存储器访问；WB写回），结构图中要有必要的控制信号与多选门，控制信号可以自行定义缩写并进行说明。并分析指令（1）执行时，在数据通路EX段中传递的数据和相应的控制信号；**

**答案与得分点：**

**数据通路如下：**

****

**lw $s3, 4($s2)的EX段：数据内容：PC+4的结果，$s2寄存器的内容，4，$s3号寄存器的编码，ALU的运算结果，Zero信号；控制信号：指令的5-0位，ALU的数据选择门ALUSrc，ALU的控制信号。（4分）**

1. **假设支持一个周期同时写读寄存器，在EX、MEM阶段有转发，若以上序列采用流水方式执行，当分支不成立，请画出执行指令（1）到（5）的流水线时空图，并计算吞吐率和加速比。（10分）**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| lw $s3, 4($s2) | IF | ID | EX | MEM | WB |  |  |  | **LW在MEM/WB转发** | |
| add $s5, $s3, $s1 |  | IF | ID | ID | EX | MEM | WB |  |  |  |
| Addi $s7, $s5, 1 |  |  | IF | IF | ID | EX | MEM | WB | **ADD在EX/MEM转发** | |
| beq $s5, $s6, Exit |  | **ADD在MEM/WB转发** | | | IF | ID | EX | MEM | WB |  |
| sw $s5, 4($s2) |  |  |  |  |  | IF | ID | EX | MEM | WB |

吞吐率=5/10=50%; 加速比=25/10=2.5.