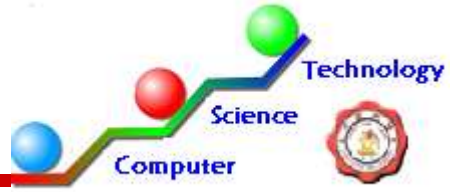
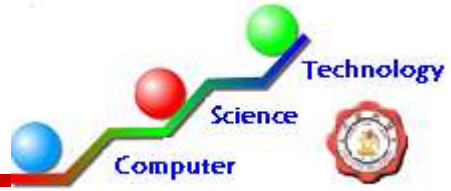


计算机组成原理第七章题解

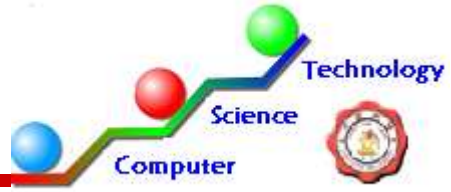
第七章 7.1



- 7.1 请分析CPU内部采用分散互连结构和单总线以及多总线结构的优缺点。

- 答：
- 分散互连结构是在需要进行数据传输的部件间设置专用通路。该结构的优势是部件间不存在传输通路的竞争问题，所以指令执行速度快。
- 单总线结构是将各部件都连接在单一的总线上。其优势是CPU结构紧凑，但由于部件间争用总线造成指令执行速度慢。
- 双总线结构是在单总线结构的基础上增加了一条总线，使得ALU的两个输入可以分别来自两条线总线。双总线结构相对于单总线结构来说，通过增加一条总线来分担数据传输流量，使得指令执行速度得以提高。
- 三总线结构是在双总线结构的基础上再增加一条总线，使得ALU的两个输入可以分别来自两条线总线，且ALU的输出连接到第三条总线上。总之，多总线结构通过增加硬件开销，换取指令执行速度。

第七章 7.2



7.2、设数据总线上接有A,B,C,D四个寄存器，要求选用合适的74系列芯片，完成下列逻辑设计：

（1） 设计一个电路，在同一时间实现A←D, B←D和C←D寄存器间的传送；

（2） 设计一个电路，实现下列操作：

T0时间完成总线← D；

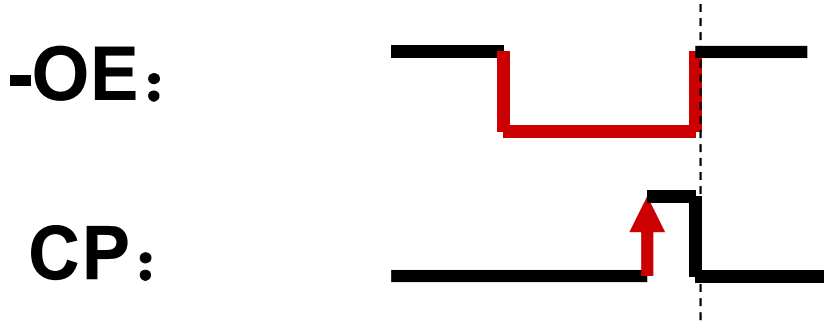
T1时间完成A ←总线；

T2时间完成总线← A；

T3时间完成B ←总线。

解：

（1）采用三态输出的D型寄存器74LS374做A、B、C、D四个寄存器，其输出可直接挂总线。A、B、C三个寄存器的输入采用同一脉冲打入。注意-OE为电平控制，与打入脉冲间的时间配合关系为：

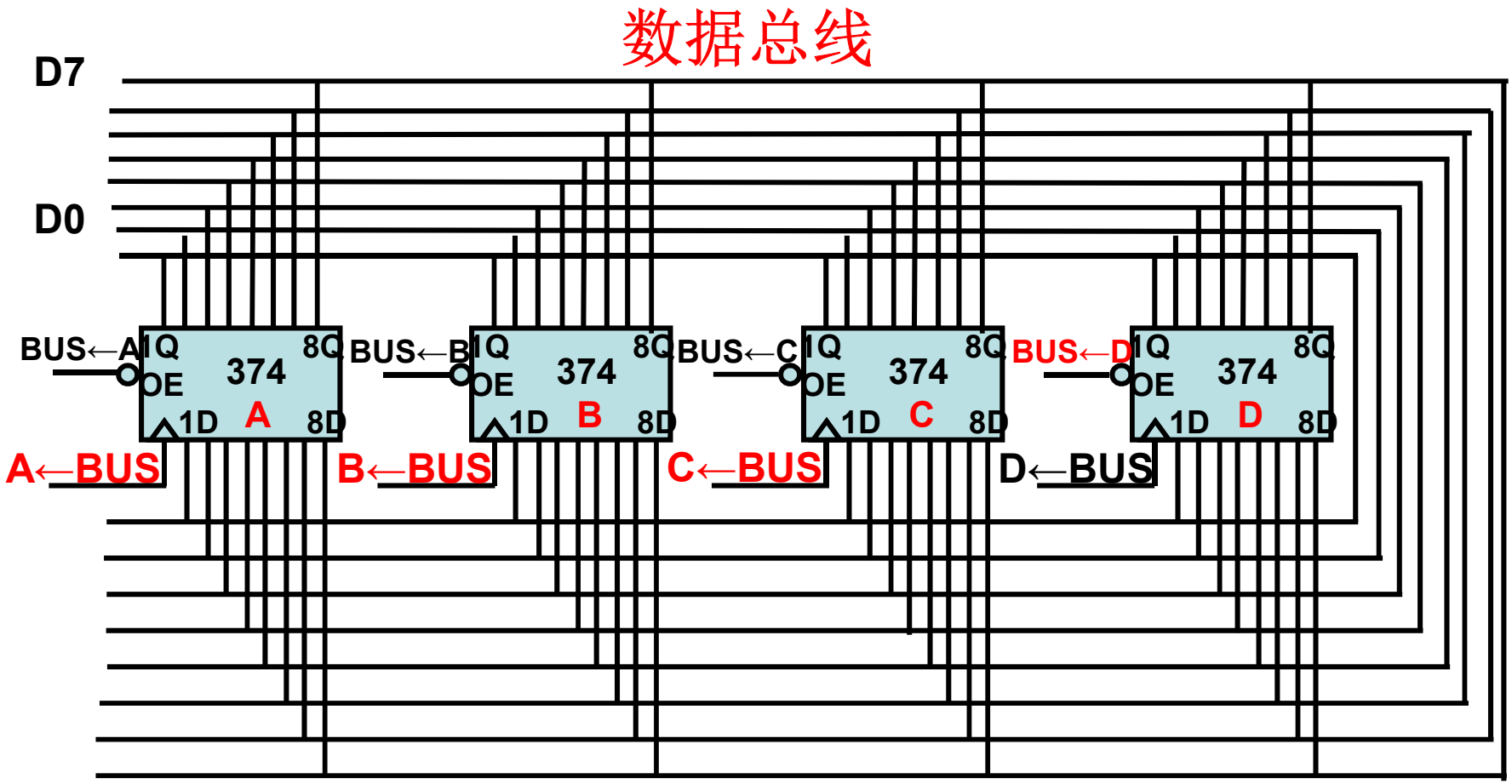


令：A←BUS=B←BUS=C←BUS=CP；

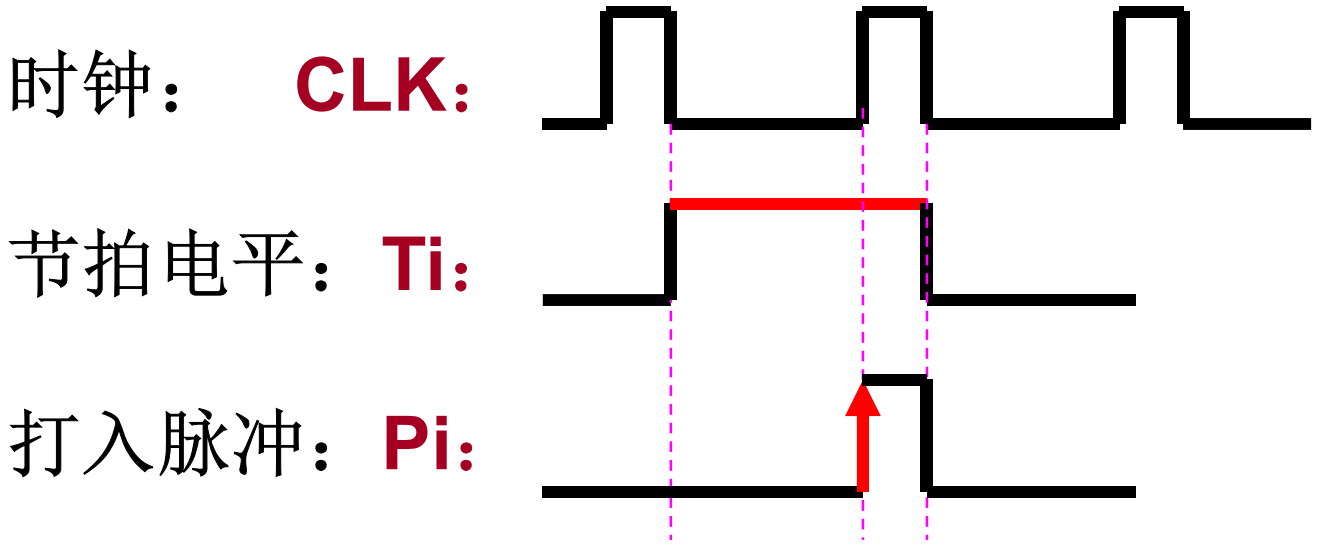
BUS←D= -OE；

当CP前沿到来时，将A、B、C←D。

现以8位总线为例，设计此电路，如下图示：

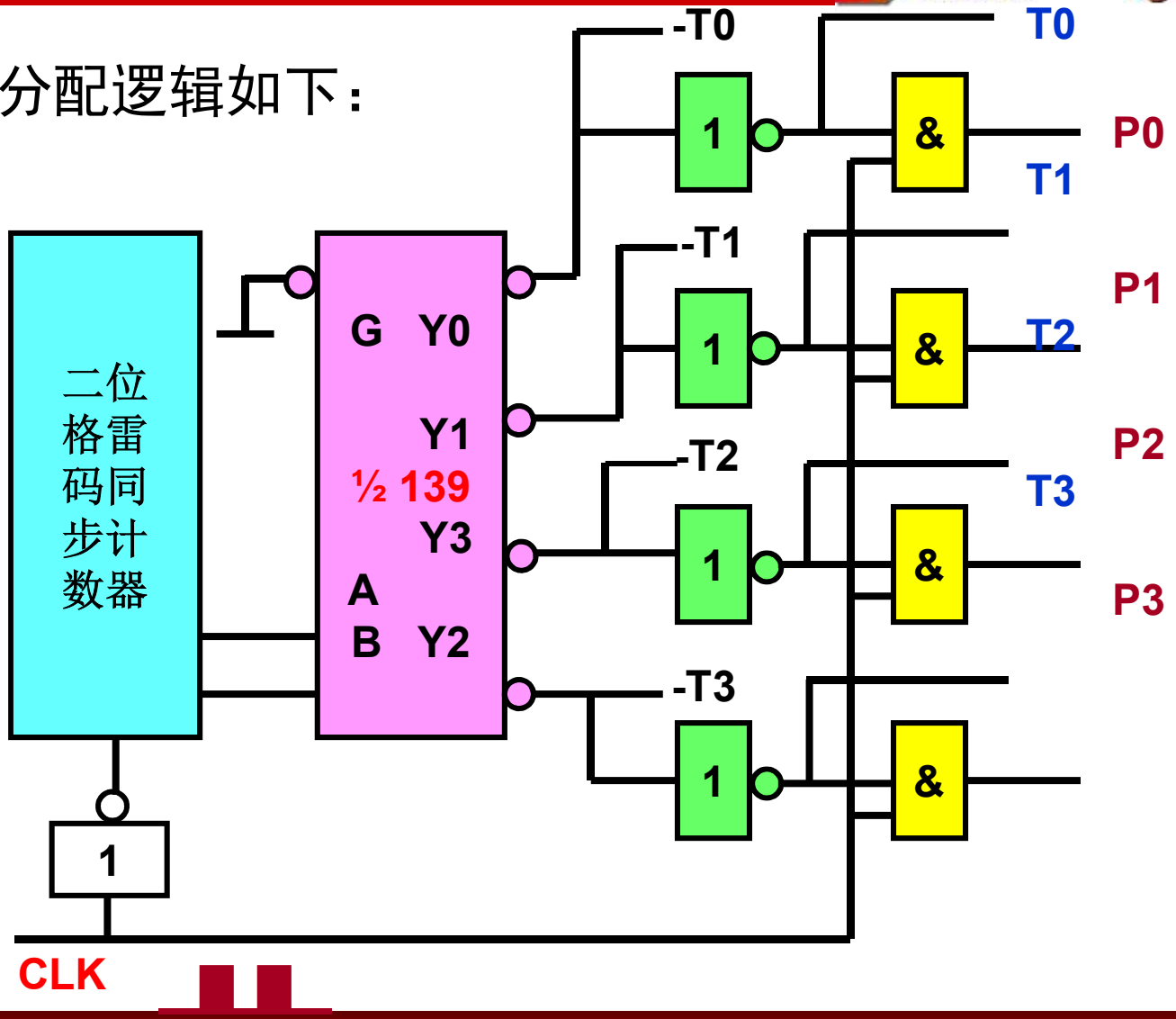


（2）寄存器设置同（1），由于本题中发送、接收不在同一节拍，因此总线需设锁存器缓冲，锁存器采用74LS373（电平使能输入）。节拍、脉冲配合关系如下：

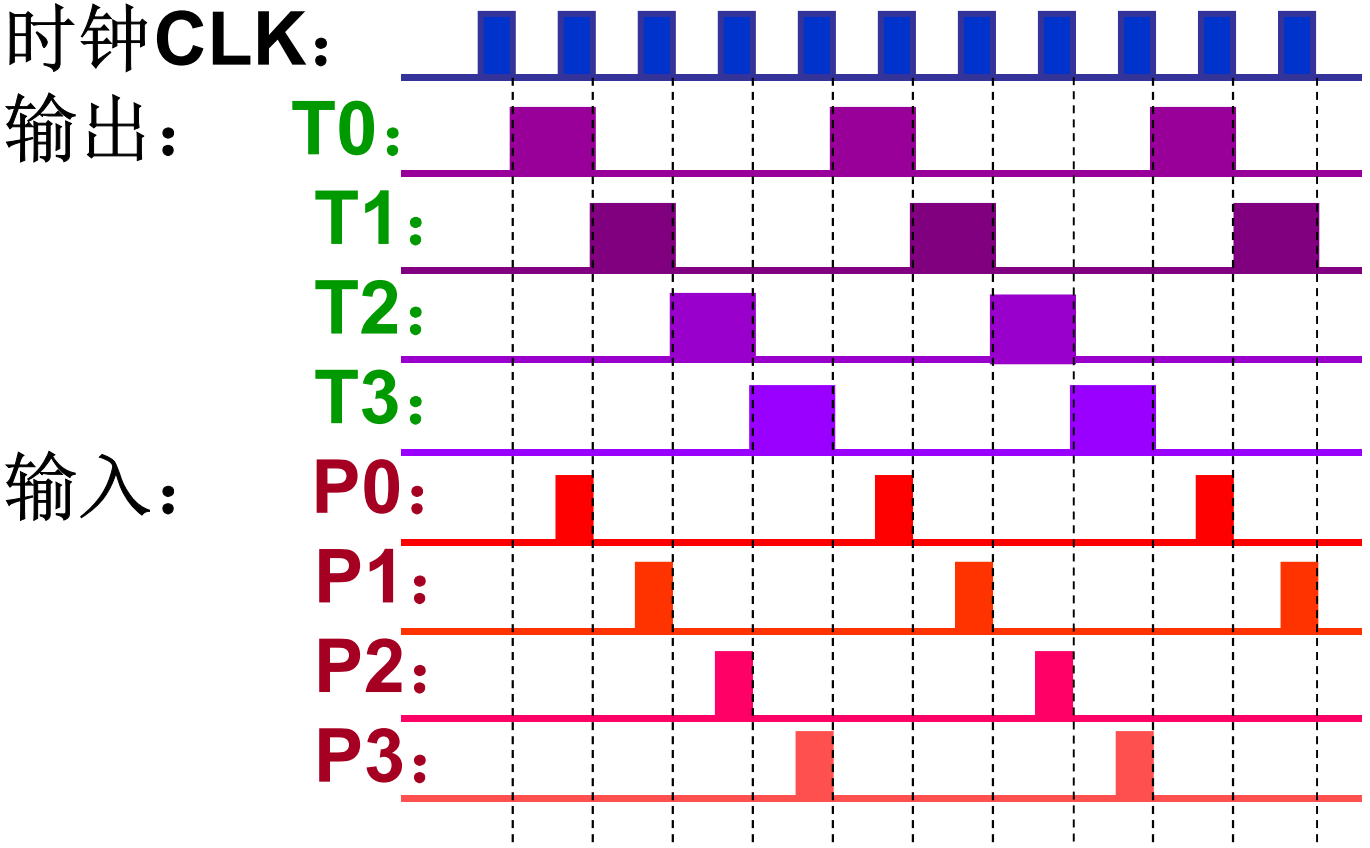


图中，脉冲包在电平中，为了留有较多的传送时间，脉冲设置在靠近电平后沿处。

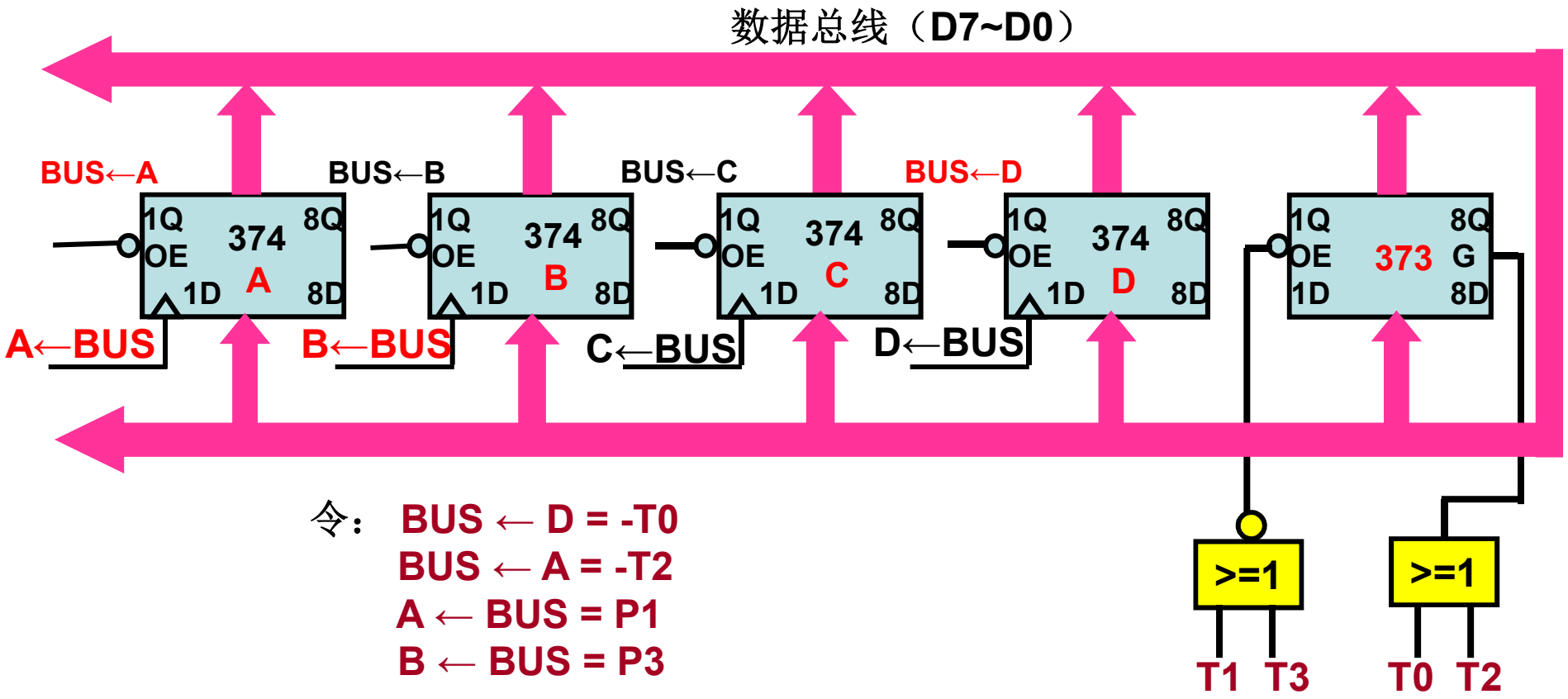
节拍、脉冲分配逻辑如下：



节拍、脉冲时序图如下：



以8位总线为例，电路设计如下：（图中，A、B、C、D四个寄存器与数据总线的连接方法同上。）



令： BUS ← D = -T0
BUS ← A = -T2
A ← BUS = P1
B ← BUS = P3

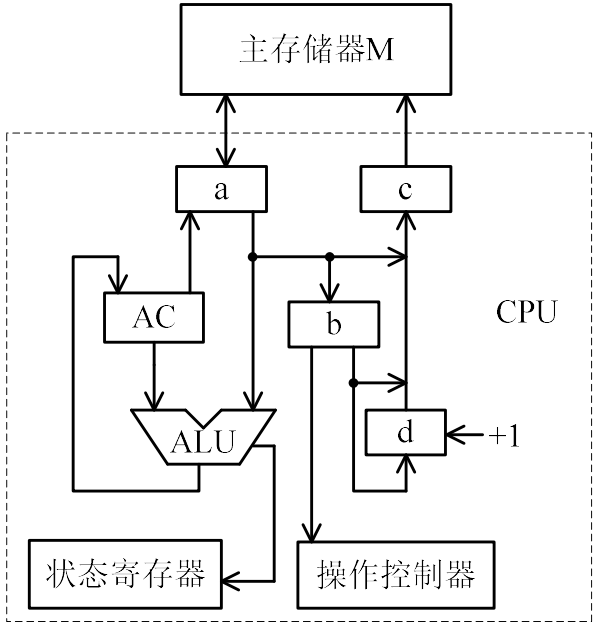
第七章 7.3

- 7.3 若某CPU的数据通路结构如下图所示，其中有一个累加寄存器AC，一个状态条件寄存器和其它四个寄存器，各部分之间连接线的箭头表示信息传送方向。要求：

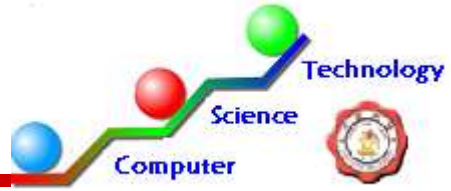
（1）写出图中a、b、c、d四个寄存器的名称；

（2）用寄存器传输语言描述指令从主存取到控制器的操作过程；

（3）设计一条加法指令，并用寄存器传输语言描述加法指令执行阶段的操作过程。

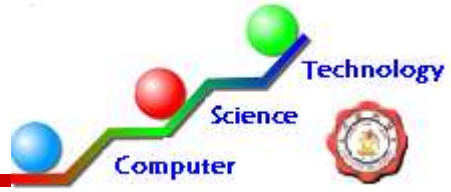


第七章 7.3



- 题解：
- (1) a ——存储器数据缓冲寄存器MDR；
  b ——指令寄存器IR；
  c ——存储器地址寄存器MAR；
  d ——程序计数器PC；
- (2) MAR← (PC);
  MDR←M[MAR] ; MemRd
  IR ← (MDR);
  PC ← (PC)+1, 操作控制器← (IR); +1

第七章 7.3



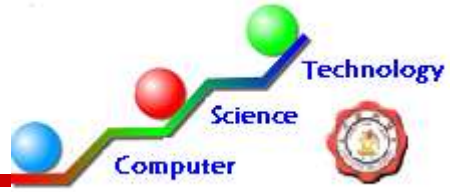
- (3) 假设加法指令为：add x，其中，x为主存单元地址。该指令的功能是将累加器AC的值与存储单元内容相加结果保存在累加器AC。

  MAR ← x;

  MDR ← M[x] ; MemRd

  ALU ← (MDR), ALU ← (AC) , AC ← ALU; +

第七章 7.4



- 7.4 欲在7.3.4给出的目标指令集中增加一条立即数加法指令addi rt,rs,imm16，若CPU采用单周期数据通路设计方案，请问在7.5.1中给出的图7-19数据通路能否支持该指令的执行？若不能，请问如何修改？并用指令周期流程图描述该指令的完整执行过程。

- 题解：
  - 支持。
  - 指令周期流程见下页。

用RTL描述指令周期流程如下：

取指阶段：

（1）addr(IM) ← (PC)；

（2）read(IM)；

（3）Add1_B ← (PC)，(PC)+4；

执行阶段：

（1）R_Reg1(RF) ← Inst [25-21]，ALU_A ←(R_data1)；

（2）W_Reg(RF) ← Inst [20-16]；RegDst=0

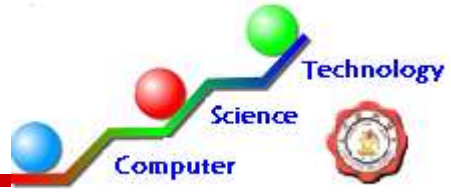
（3）SigExt16/32 ← Inst [15-0]，ALU_B← SigExt16/32；ALUSrc=1

（4）ALU操作（add）；　　ALUOp=00，ALUCtrl=100

（5）W_data(RF) ← ALU_C，PC ← (PC)+4；

Branch=0，Jump=0，MemtoReg=0，RegWr=1，CLK↑

第七章 7.5



- 7.5 在7.6.3中给出的图7-24多周期数据通路中，是否可以不要指令寄存器IR，而直接对存储器数据寄存器MDR中的信息进行译码？为什么？

- 答：
  - 不可以。
  - 因为若不设置IR的话，从存储器读出的指令和数据都暂存于MDR。Load指令在第四个时钟周期读出的数据将覆盖掉指令，导致指令后续阶段无法按照指令要求执行相应操作。

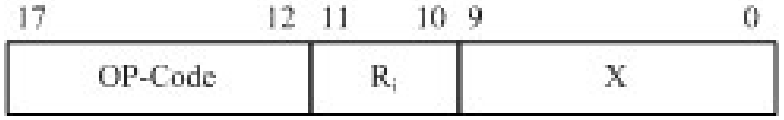
第七章 7.6

- 7.6下图所示的CPU逻辑框图中，有两条独立的总线和两个独立的存储器。已知指令存储器IM最大容量为16384字（字长18位），数据存储器DM最大容量是65536字（字长16位）。各寄存器均有“打入”（Rin）和“送出”（Rout）控制命令，但图中未标出。

  （1）指出下列各寄存器的位数：

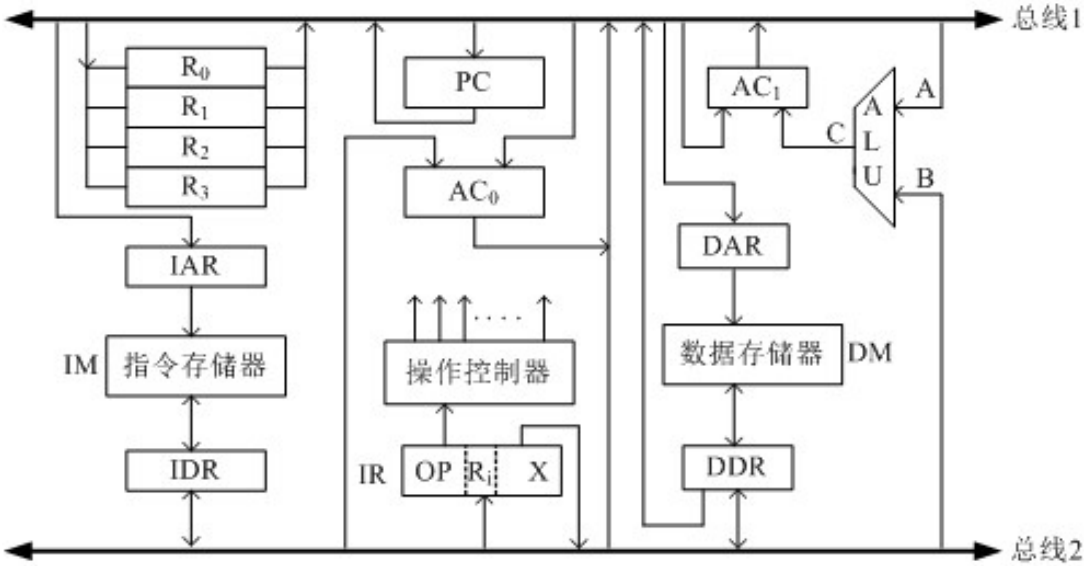
  程序计数器PC，指令寄存器IR，累加器AC_0和AC_1，通用寄存器R_0～R_3，指令存储器地址寄存器IAR，指令存储器数据寄存器IDR，数据存储器地址寄存器DAR，数据存储器数据寄存器DDR；

  （2）设该CPU的指令格式为：

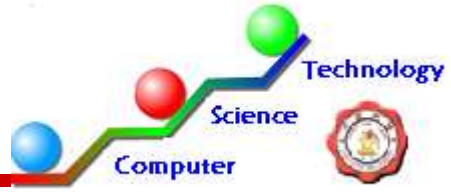


加法指令可写为“ADD X（R_i）”，其功能是：

$AC_1 \leftarrow$（AC_0）+（（R_i）+X），其中（（R_i）+X）部分通过寻址方式指向数据存储器，用指令周期流程描述ADD指令从取指令开始到执行结束的操作过程，并标明完成具体操作所需要的微操作控制信号。



第七章 7.6



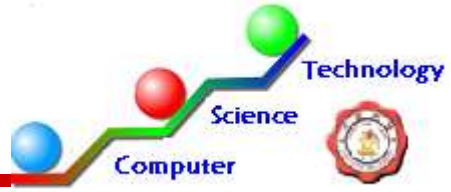
- 题解：

(1) PC = IAR = 14位；IR = IDR = 18位；

$AC_0 = AC_1 = R_0 \sim R_3 = DDR = DAR = 16$位

(2) 接下页

第七章 7.6



取指令阶段：

（1） IAR ← (PC)； PCout，IARin

（2） IDR ← IM[IAR]；IM读，IDRin

PC ← (PC) + 1； PC+1，PCin

（3） IR ← (IDR)； IDRout，IRin

指令执行阶段：

（1） ALU_A← (R_i)，ALU_B ← (IR)X；Riout，(IR)Xout, Add

AC1 ←ALU_C； AC_1in

（2） DAR ← (AC1)； AC_1out，DARin

（3） DDR ← DM[DAR]； DM读，DDRin

（4） ALU_A ← (AC0)，ALU_B ← (DDR)；AC_0out，DDRout, Add

AC1 ←(ALU_C)； AC_1in

注：分号左边为微操作，分号右边为所需微命令（微操作控制信号）

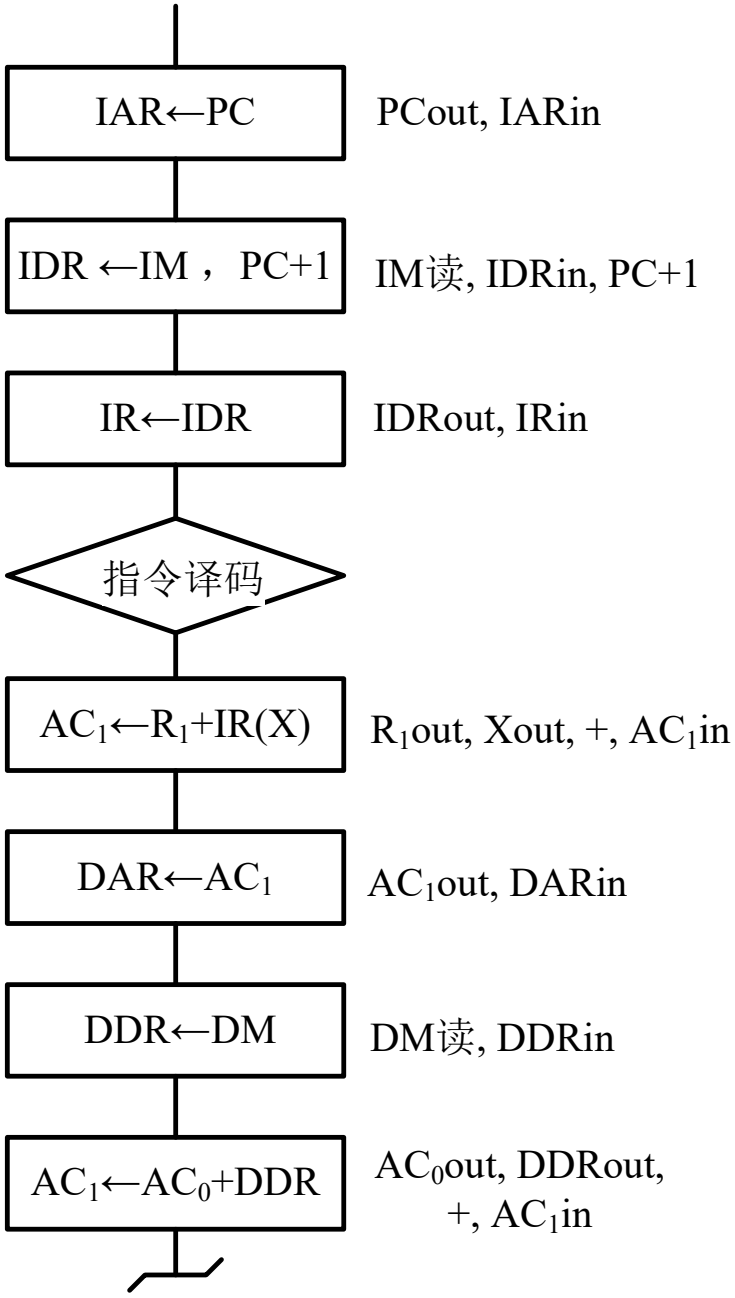
第七章 7.6

解法2：

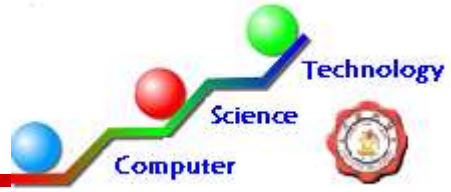
用指令周期流程图描述。

左边为指令周期流程图，

右边为所需微命令信号序列。



第七章 7.7



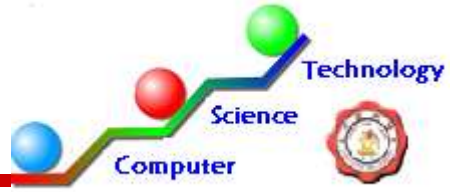
- 7.7 某CPU的主频为4MHz，各类指令的平均执行时间和使用频度如下表所示。

指 令 类 别	存取	加、减、比较、转移	乘除	其它
平均指令执行时间	0.6μs	0.8μs	10μs	1.4μs
使用频度	35%	50%	5%	10%

（1）试计算该CPU的速度（单位用MIPS表示）；

（2）若上述CPU主频提高为6MHz，则该CPU的速度又为多少？

第七章 7.7



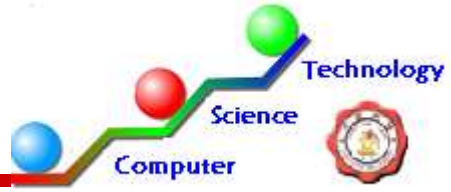
- 题解：

(1) 平均指令执行时间 = 0.6 × 35% + 0.8 × 50% + 10 × 5% + 1.4 × 10% = 1.25 μs

CPU速度=1/1.25μs=0.8MIPS

(2) 0.8 × (6/4) = 1.2MIPS

第七章 7.8



- 7.8 欲在7.6.1给出的图7-24多周期数据通路上执行一条类似IA-32的加法指令“ADD AX，BX（100）”，该指令的功能是将BX内容加上100的值作为地址读取存储器，再将AX的内容与存储器单元内容相加，结果送AX。要求：

  （1）按照MIPS 32指令格式，设计该加法指令的格式。

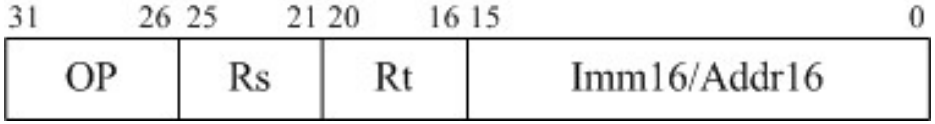
  （2）为了实现该加法指令，请说明如何修改图7-24给出数据通路。

  （3）用指令周期流程图描述该加法指令的完整执行过程。

第七章 7.8

❒ 解：

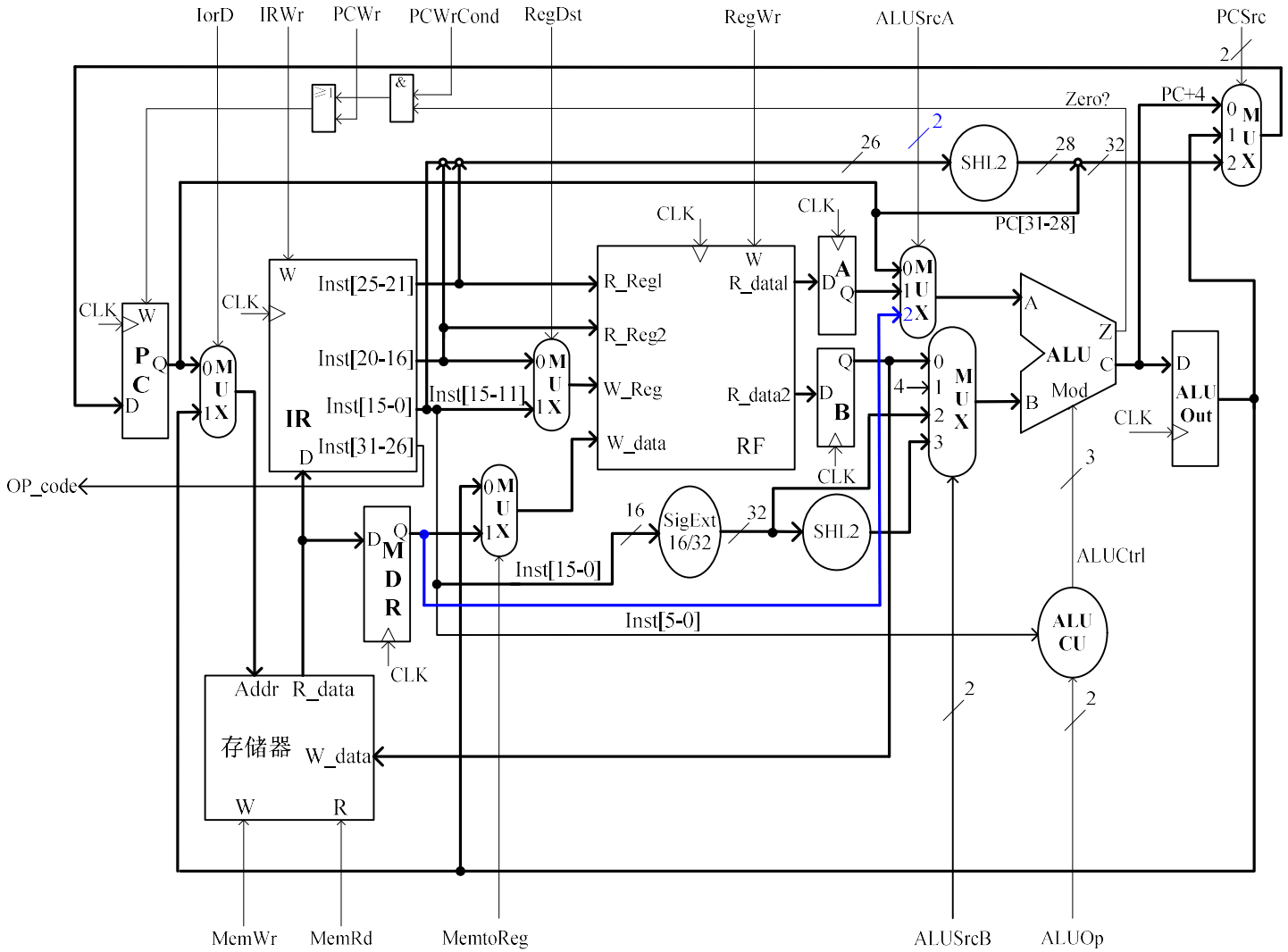
(1) 可以借用MIPS 32 I-型指令格式：

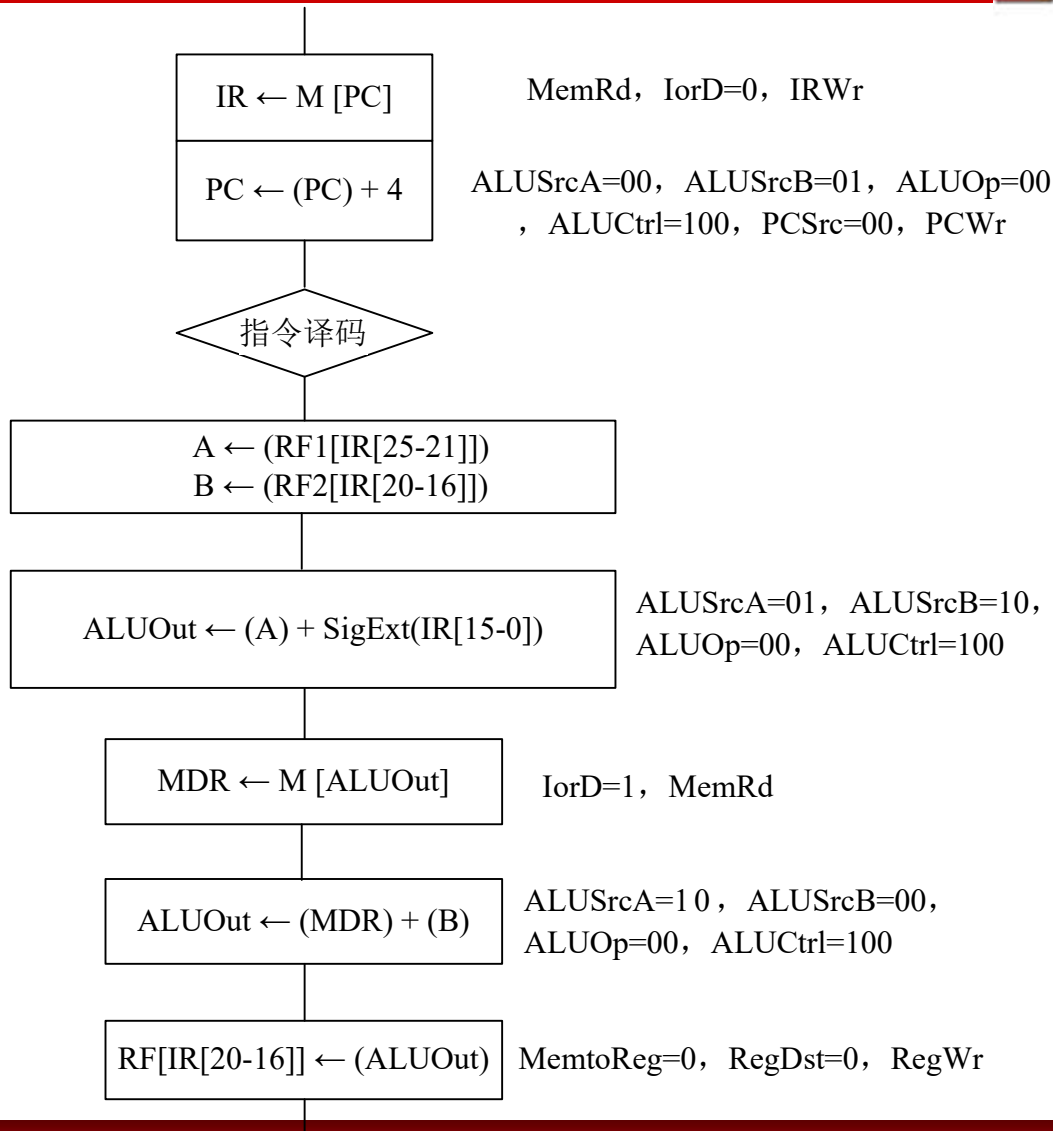


（2）修改后的数据通路图如下页： (蓝色部分)

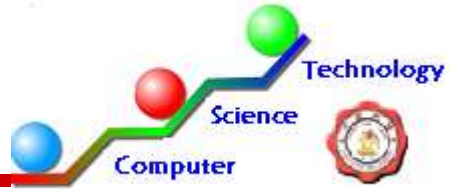
（3）指令周期流程图如下下页：

第七章 7.8



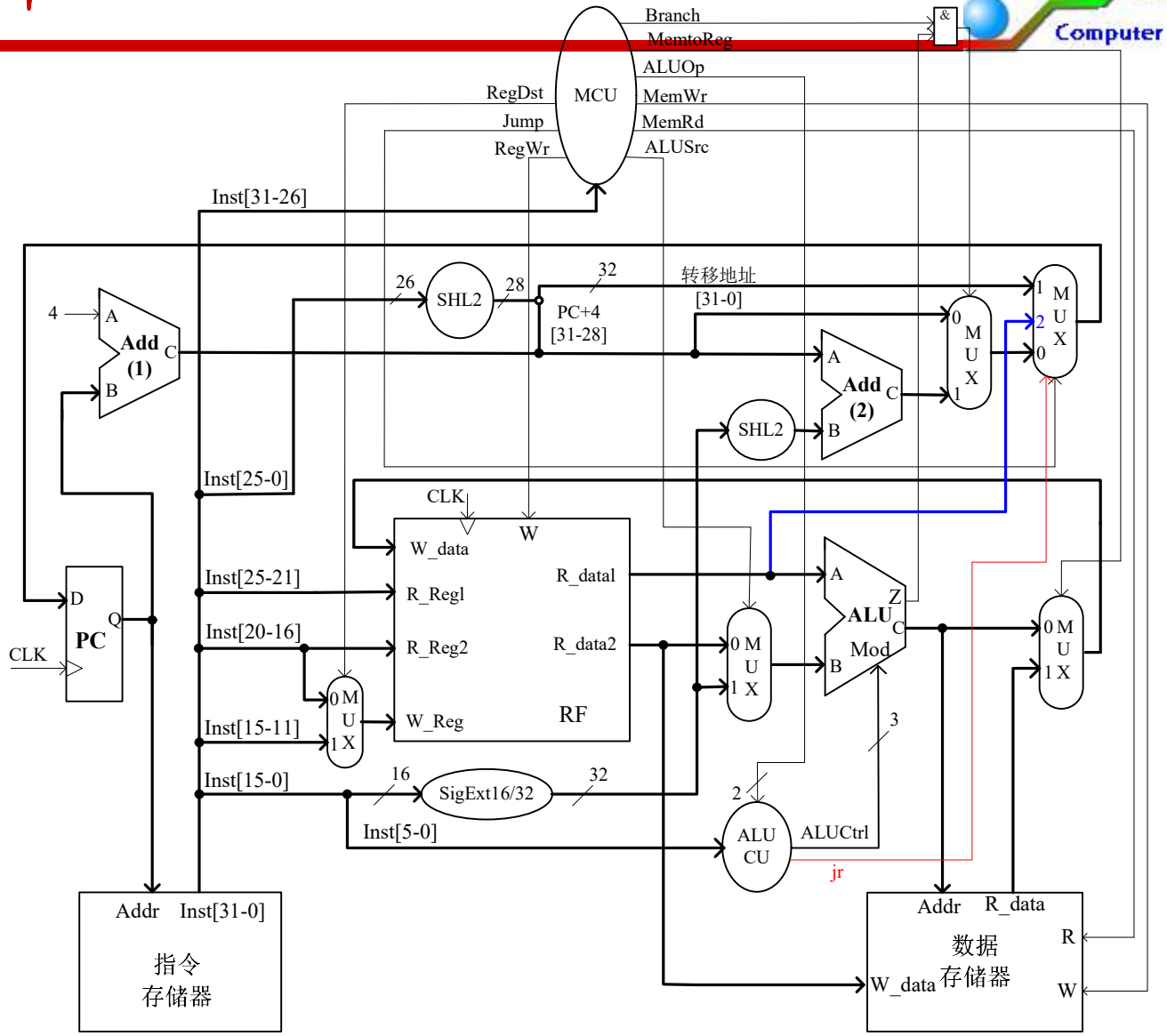


第七章 7.9

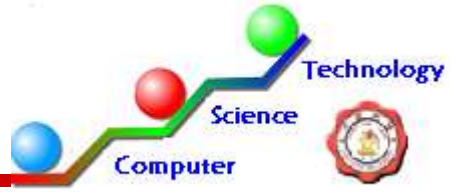


- 7.9 欲给本章描述的单周期数据通路加入 jr Rs指令（其功能是按照寄存器 Rs 内容转移。指令格式属于R-型，其中Func字段为“001000”），请在图7-19给出的单周期数据通路加入必要的数据通路和控制信号。
- 解：修改后的单周期数据通路如下页: (彩色部分)

  ALUCU输出增加了一个控制信号jr,若ALUOP=00，Func=001000时，jr=1；当jr=1，Jump=0时，按照读出寄存器的内容转移。



第七章 7.10



- 7.10 考虑一个恒0错误（某个控制信号的值始终保持为0），对图7-19的单周期数据通路的影响。分别考虑当RegWr、ALUOp、Branch、MemRd和MemWr信号发生恒0错误时，在7.3.4给出的目标指令中，哪些指令仍能正常工作？为什么？
- 解

RegWr恒0时，sw,beq,j指令仍能正常工作。因为这3条指令不涉及寄存器写操作。

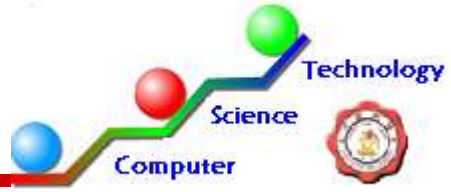
ALUOp恒0时，lw,sw,add指令仍能正常工作。执行lw,sw指令时ALUOp=00；执行add指令时要求ALUCtrl=100，而ALUOp=00 时恰好ALUCtrl=100 。

Branch恒0时，除了beq之外其它指令都仍能正常工作。

MemRd恒0时，除了lw之外其它指令都仍能正常工作。

MemWr恒0时，除了sw之外其它指令都仍能正常工作。

第七章 7.11

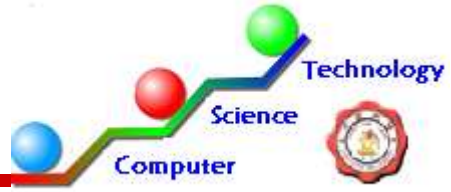


- 7.11 若想在7.3.4中给出的目标指令集中增加一条lw指令的变形指令：l_inc，该指令的功能相当于下面2条MIPS 32指令，即从存储器读取数据后，将存储单元地址自增1。请在图7-19的单周期数据通路加入必要的数据通路和控制信号。

```
lw   Rt, l(Rs)
addi Rs, Rs, 1
```

- 解：要实现这条指令，对图7-19的修改比较复杂。主要修改包括：
  1）将2R+1W RF换成2R+2W RF；
  2）对Add 2和2R+2W RF的输入、输出进行修改，实现
  Rs ←(Rs)+1。

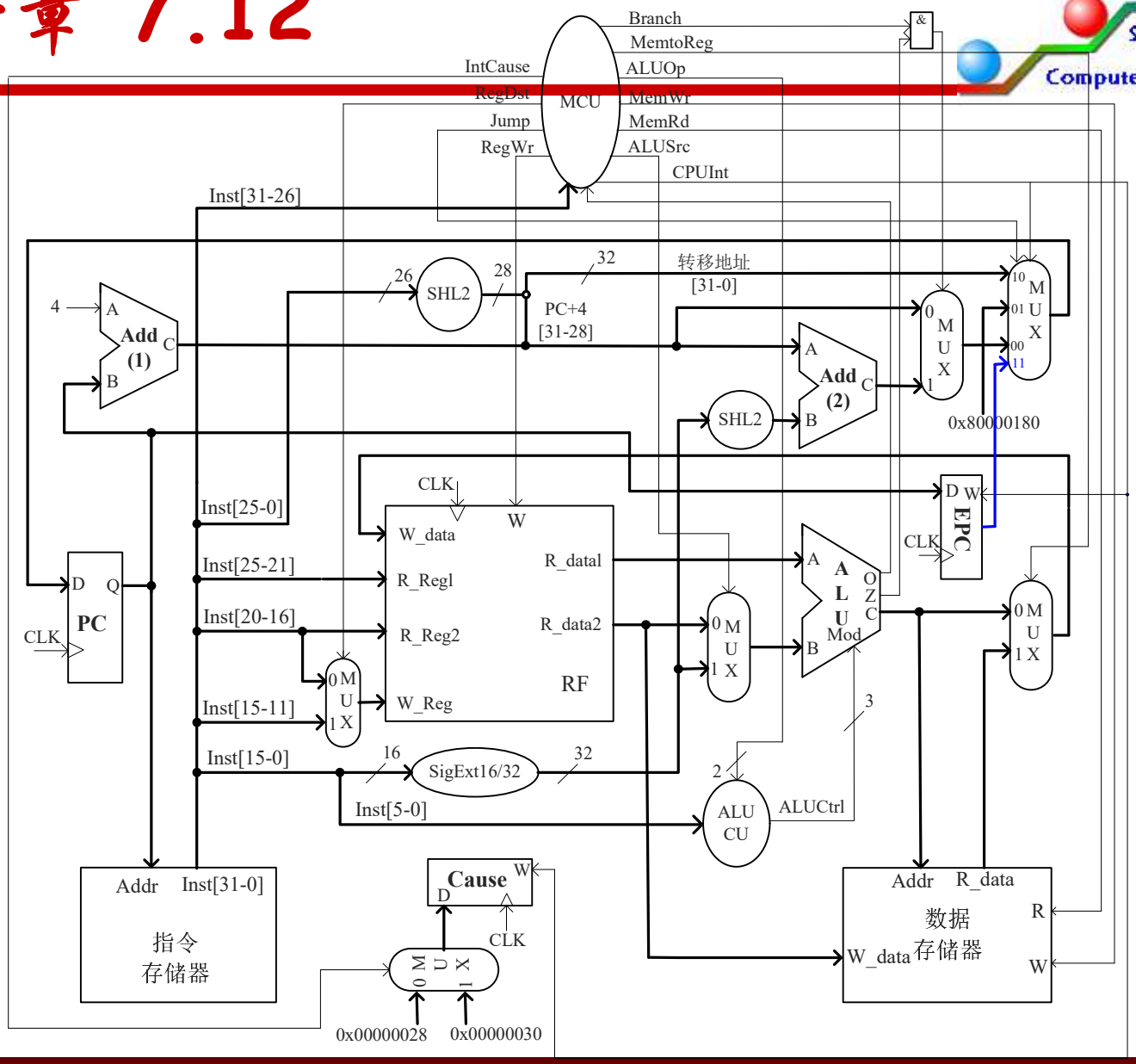
第七章 7.12



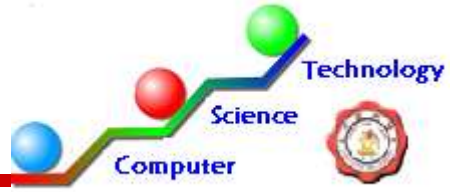
- 7.12 若想在7.3.4中给出的目标指令集中增加一条eret指令（异常返回指令），该指令的主要功能是将发生异常时的指令地址重新存入PC。请说明如何修改图7-22给出的数据通路使其支持该指令。

- 解:对图7-22数据通路修改如下页: (蓝色部分)

  执行该指令时, MCU发出 Jump=1，CPUInt=1控制信号。

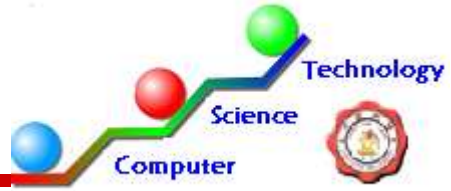


第七章 7.13



- 7.13 在MIPS系统中，操作系统如何识别异常中断的原因以及如何转入中断服务程序？请针对图7-22给出的数据通路讨论。
- 解:在MIPS系统中，当异常发生时在异常程序计数器（EPC）中保存出错指令的地址，并把控制权转交给特定地址（0x8000 0180）处的操作系统程序。操作系统程序读取Cause寄存器的值，识别中断源后设置新的PC值，使CPU转移到异常处理程序对异常进行相应处理。在图7-22中要提供对Cause寄存器访问的数据通路，并在指令系统中设计一条管态指令，该指令由操作系统用来读取Cause寄存器值到一个通用寄存器，然后通过条件转移指令实现异常原因识别及转移到异常处理程序。

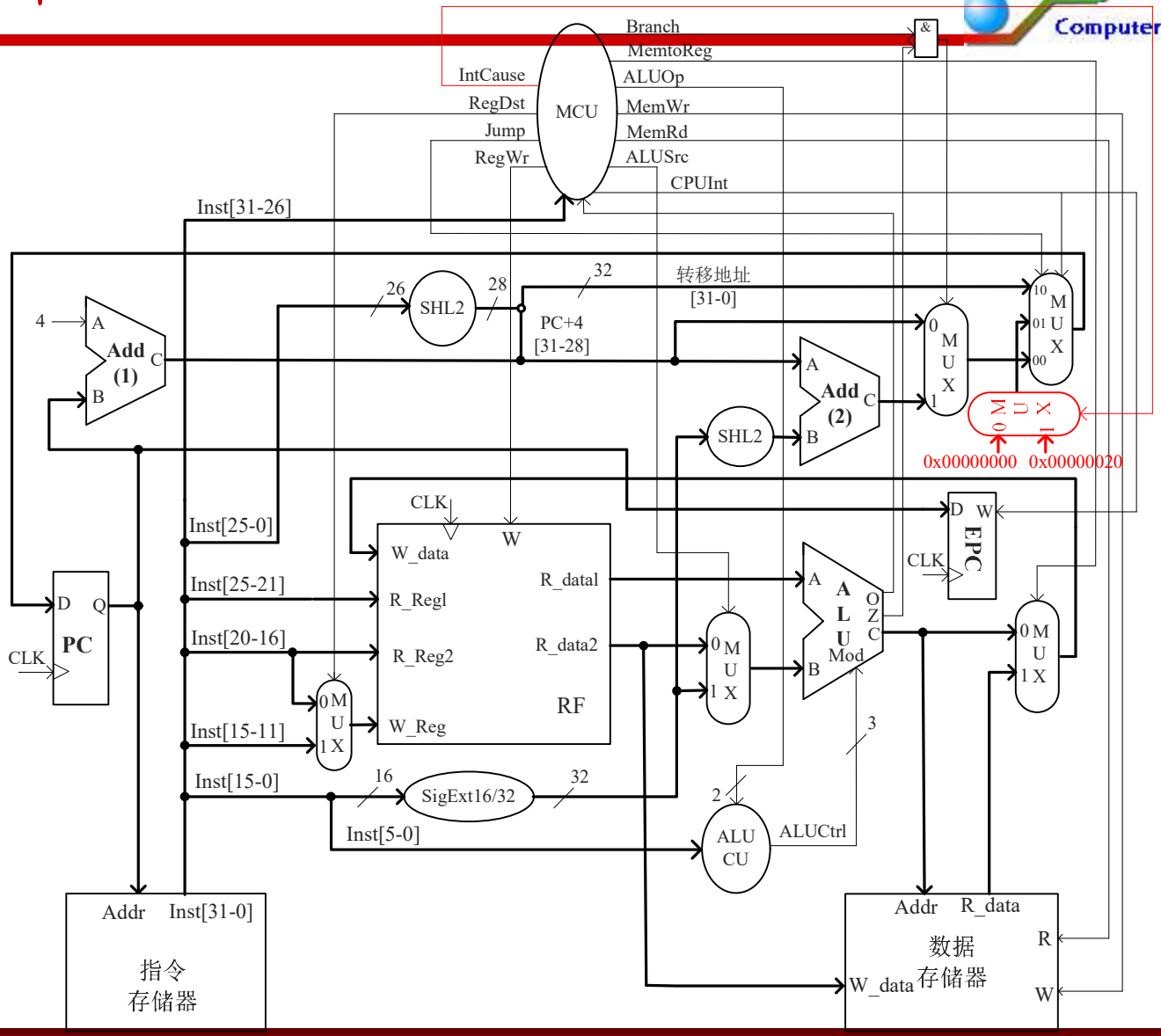
第七章 7.14



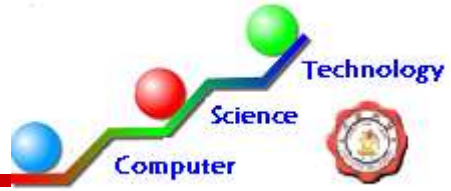
- 7.14 在MIPS系统中，若采用硬件向量中断，如何修改图7-22给出的数据通路实现异常中断原因识别和转入中断服务程序？

- 解:假设仅支持未定义指令异常和算术溢出异常，为了简化设计，假设未定义指令异常向量地址为0xc000 0000，算术溢出异常向量地址为0xc000 0020。那么，对图7-22数据通路修改如下页: (红色部分)

第七章 7.14



第七章 7.16

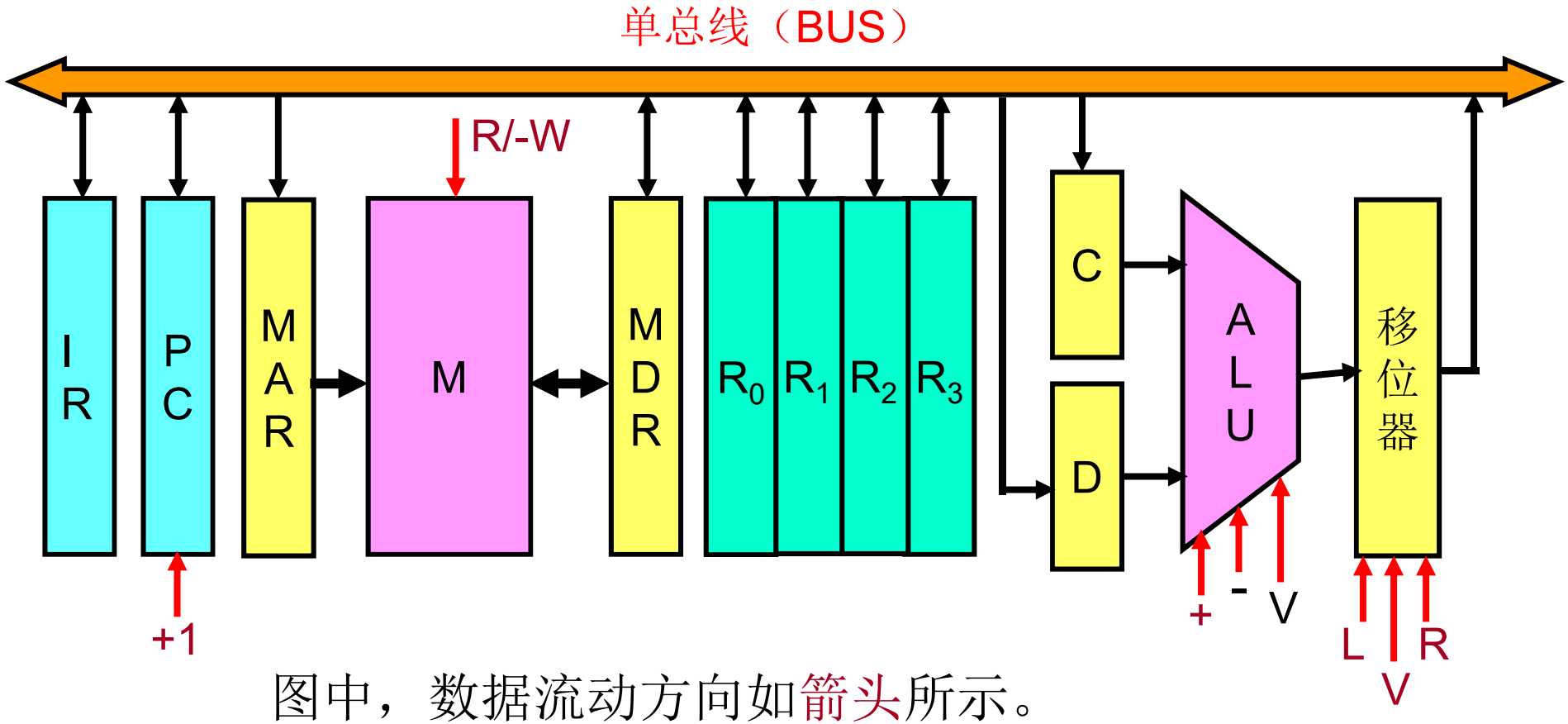


- 7.16 某计算机CPU中有如下部件：ALU（具有+、-、直送V等功能），移位器（具有左移L、右移R、直送V等功能），主存储体M，主存数据寄存器MDR，主存地址寄存器MAR，指令寄存器IR，程序计数器PC（具有自增+1功能），通用寄存器R0～R3，暂存器C（连ALU左输入端）、D（连ALU右输入端）。

  (1) 若上述逻辑部件按单总线结构组成CPU数据通路，请画出指令“SUB （R0），R3”的指令周期流程图，该指令的含义为：源操作数的有效地址在寄存器R0中，目的操作数在R3中，目的操作数减去源操作数后结果送到目的操作数的有效地址中；（注：减法时，要求被减数送ALU的左输入端，减数送ALU的右输入端）

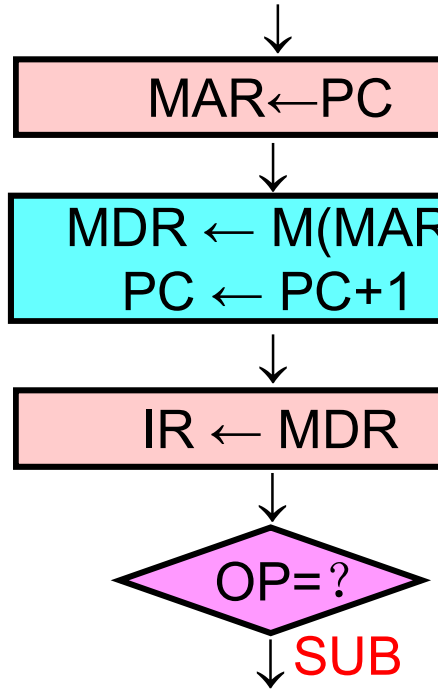
(2) 若用类似MDRo的方式表示寄存器输出类微命令，类似MDRi的方式表示寄存器输入类微命令，1→R表示主存读命令，1→W表示主存写命令，MDR、MAR与主存储体M之间，暂存器C、D与ALU输入端之间是直通的，不需要微命令控制。请写出对应该指令周期流程图所需的全部微操作命令序列。

解：（1）采用单总线结构的CPU硬件框图如下：



图中，数据流动方向如箭头所示。

(1) SUB(R0)，R3指令流程图　　(2) 对应的全部微操作命令

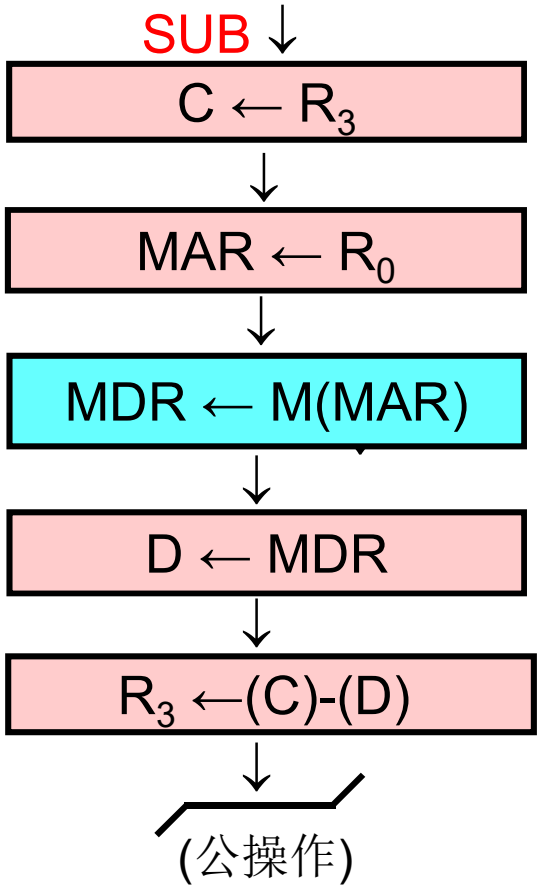


PC_o，MAR_i

$1 \rightarrow R$，+1

MDR_o，IR_i

(1)



(2)

$R3_o$，C_i

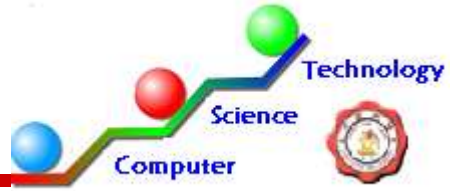
$R0_o$，MAR_i

1→R

MDR_o，D_i

−，V，$R3_i$

第七章 7.18

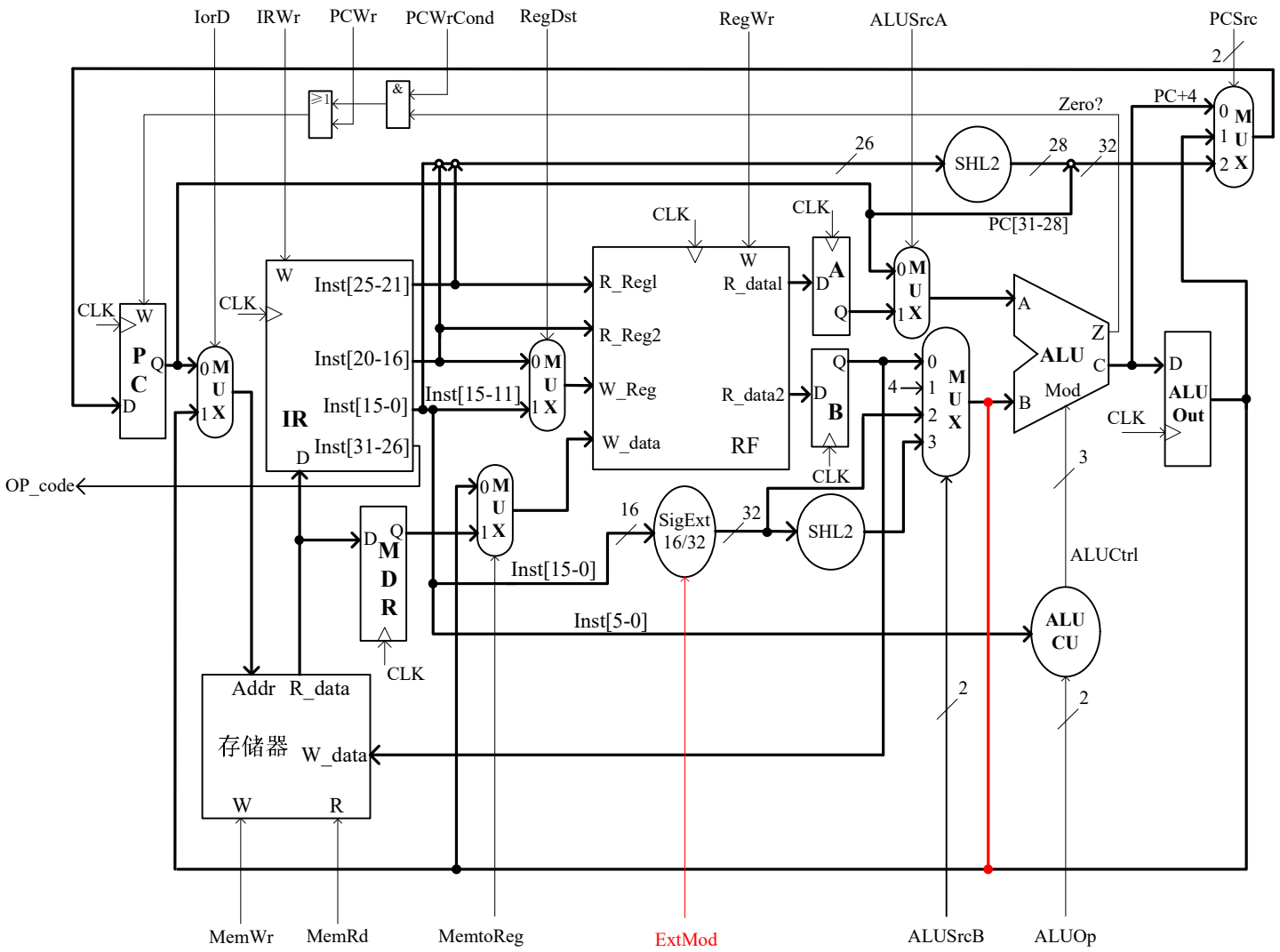


- 7.18 在图7-24多周期数据通路的基础上，通过添加适当的通路和控制信号，实现取立即数指令lui Rt，imm16（I-型指令格式，其功能是将立即数读入寄存器高16位）。请用RTL描述该指令周期流程。

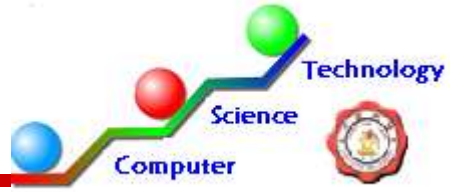
- 解:对图7-24数据通路修改如下页: (红色部分)

  对原来符号扩展部件（SigExt16/32）功能进行扩展，增加将16位立即数放入高位、低位补16位全零的功能。增加控制信号（ExtMod）控制在符号扩展和将立即数读入高16位之间进行功能选择。

第七章 7.18



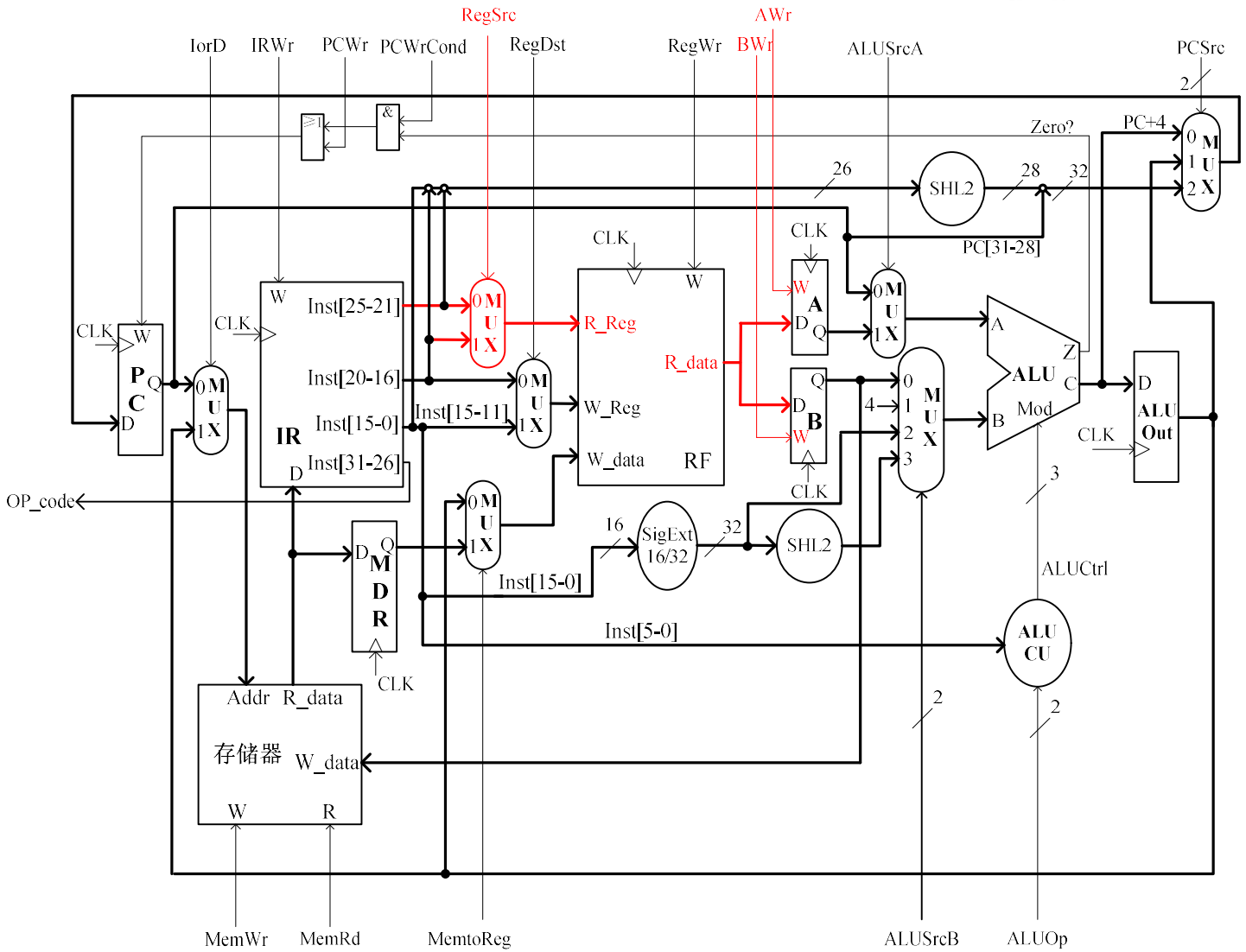
第七章 7.19



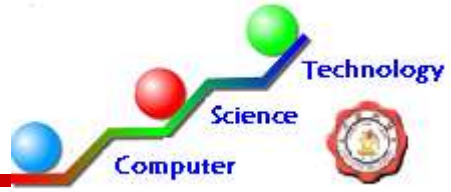
- 7.19 若将多周期CPU实现中的寄存器堆RF改为只有一个读端口，用图描述此修改所带来的数据通路的其它必要修改，并分析对指令周期的影响。
- 解:对图7-24数据通路修改如下页: (红色部分)

  由于2个寄存器不能在同一个时钟周期内同时读出，而是在连续2个时钟周期内读出，所以，指令周期延长一个时钟周期。在前一个时钟周期内，RegSrc=0,AWr=1,读出Rs寄存器内容打入A暂存器；在后一个时钟周期内，RegSrc=1,BWr=1,读出Rt寄存器内容打入B暂存器；

第七章 7.19



第七章 7.20



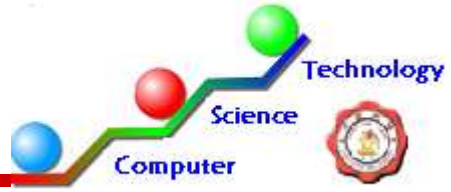
- 7.20 由于指令周期=CPI×时钟周期，所以主频和CPI是影响CPU性能的两个重要参数。但是，同时降低这两个参数往往是不可能的，在设计过程中始终存在这两个参数的权衡。具体方法如下：

  （1）以提高CPI为代价，提高处理器的主频；

  （2）降低主频换取较小的CPI。

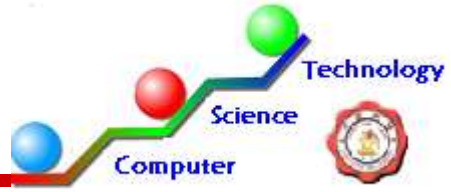
  请分析MIPS多周期数据通路，分别说明这两种方法实现的具体措施。

第七章 7.20



- 解答：
- MIPS多周期CPU数据通路的设计思想是：将指令执行过程分成时间大致相同的多个阶段，每个阶段完成少数几个微操作，各阶段的最长时间设置为一个时钟周期。
- （1）提高CPI，即增加时钟周期数。这种方法的具体措施是把指令执行过程划分成较多的阶段，每个阶段完成较少的微操作。
- （2）降低主频，即延长时钟周期。这种方法的具体措施是把指令执行过程划分成较少的阶段，每个阶段完成较多的微操作。

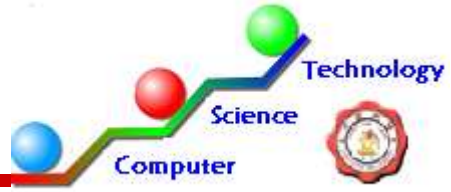
第七章 7.21



- 7.21 有三台计算机M1、M2和M3，CPU主频分别为：3.2GHz、2.8GHz、1GHz。假设这三台机器可以运行同一个指令系统，该指令系统由三类指令构成，各类指令在程序中的使用频度以及在三个CPU中实现的CPI如下表所示。则每台计算机的运行速度分别是多少？

指令类型	使用频度	CPI		
		M1	M2	M3
存数/取数	50%	5	4	3
运算	35%	4	3	3
其它	15%	3	3	3

第七章 7.21



- 解答：
- M1：

  平均CPI=50%*5+35%*4+15%*3=4.35

  平均速度=1/(4.35*1/3.2G)=736MIPS
- M2：

  平均CPI=50%*4+35%*3+15%*3=3.5

  平均速度=1/(3.5*1/2.8G)=800MIPS
- M3：

  平均CPI=50%*3+35%*3+15%*3=3

  平均速度=1/(3*1/1G)=333MIPS