# 计算题

## 1. CPU 时间的计算——给定两台计算机的 CPI 和 cycle time，进行性能的比较

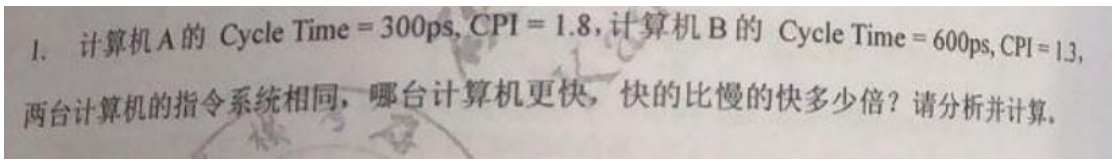
**知识点：**

**CPU Time = clock cycles × Cycle time = clock cycles/clock rate**

**= I(指令数)×CPI × Cycle time**

**CPI: 每条指令所需的时钟周期数的平均值 = clock cycles/I(指令数)**

**例题：**




2. 假定处理器时钟率为每秒 2GHz,其对应的 CPI 为 4,如果一个程序执行的时间为 20 秒，那么执行该程序的时钟周期和指令数分别是多少？。

答：40G, 10G

3. Computer A 的 Cycle Time = 250ps, CPI= 2.0；
Computer B 的 Cycle Time = 500ps, CPI =1.2;
采用相同的 ISA,则计算机的运行速度关系为是？。

答：Computer A 是 Computer B 的 1.2 倍

4. 计算机 A 的 Cycle Time =200ps, CPI = 1.5, 计算机 B 的 Cycle Time=400ps, CPI=1.0。两台计算机采用同样的 ISA,哪台计算机更快?快多少?需要给出分析计算过程。

答：设指令数为 I，则 CPU 时间：

A: CPU Time = I*CPI*Cycle Time = I*1.5*200=300Ips

B: CPU Time = I*1.0*400 = 400Ips

故 A 计算机更快

B/A = 400I/300I = 4/3=1.33

A 是 B 的 1.33 倍

~~2. 计算二进制浮点数加减法，以此说明 mips 计算机进行浮点数加法的主要步骤。~~

2.1 举例：以 $x=2^{01}\times 0.1101, y=2^{11}\times(-0.1010)$, 求$x+y$为例

两数用补码表示，阶码和尾数均采用双符号位

$[x]_{补}$ = 0001 **00**.1101 $[y]_{补}$ = 0011 **11**.0110

①零操作数检查

②对阶：小阶向大阶对齐

由于 0001-0011 = 1110 = -2 (补码计算),故 x 向 y 对齐得：

$[x]_{补}$ = 0011 **00**.0011 $[y]_{补}$ = 0011 **11**.0110

③尾数相加：$[x+y]_{补}$ = 0011 11.1001

④规格化：0010 11.0010 左归

**注意：**

规格化数：原码：正数：0.1..... 负数：1.1.....

补码：正数：0.1..... 负数：1.0.....

舍入处理：0 舍 1 入法，恒置 1 法，就近舍入等

尾数为原码时：

①最高位为 1 **或者** 移出的几位中有 1 的数值位，低位置 1

②0 舍 1 入：丢失的最高位是否为 1

尾数为补码时：

①正数与前相同

②仅当丢失位为 1 **并且** 移出的几位中有 1 的数值位才低位置 1

⑤判断溢出与否：例题中显然没有

**注意：**

溢出判断方法

当阶码用补码表示时：01,.....上溢，10,.....下溢 (双符号位)

当阶码用移码表示时：10.....正上溢，11.....负下溢 (在补码的基础上加 1 即可)

其他：$[x+y]_{移}=[x]_{移}+[y]_{补}$,01 正数，00 负数

**移码就是补码的符号位取反即可**

2.2 乘除法：乘法[阶码相加，尾数相乘], 除法[阶码向减，尾数相除]，具体自己看书或 PPT

2.3 转化为 IEEE754 单精度格式

S(符号位 1 位) E(阶码：8 位) M(尾数: 23 位)

$(-1)^s\times(1+M)\times2^E=(-1)^s\times(1.M)\times2^{E-127}$（E机器码）

2.3.1 十进制转 IEEE754 步骤：

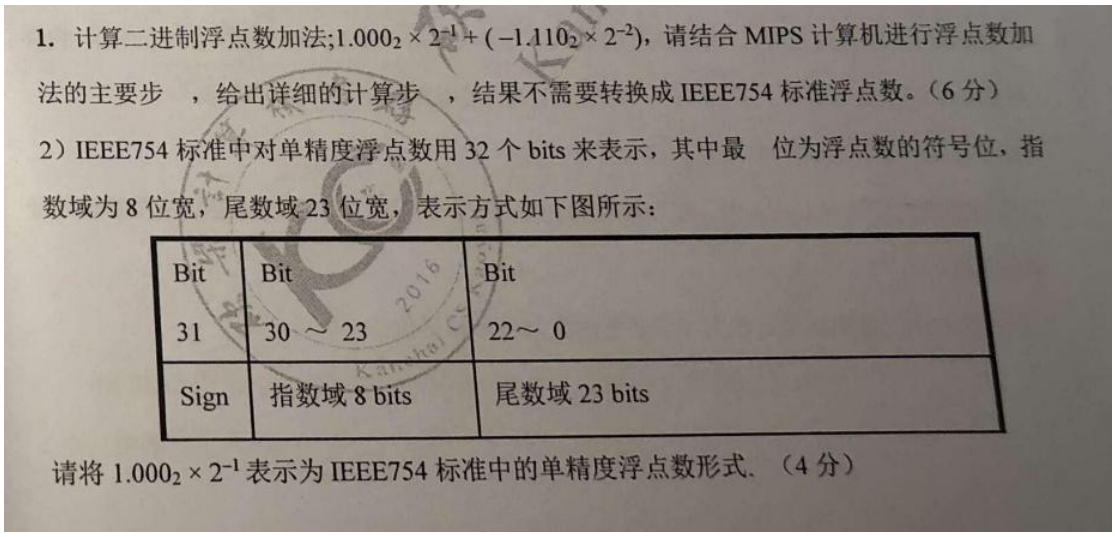
转化为二进制----变为 1.M × 2^e ---- E=e+127----写成 SEM 格式

2.3.2 IEEE754 转十进制

找 S,E,M 对应位数----写成标准格式-----转换为二进制-----转为 10 进制

双精度：S：1 E：11 M：52 e=E-1023

**练习题：**



1. 计算二进制浮点数加法;$1.000_2 \times 2^{-1} + (-1.110_2 \times 2^{-2})$，请结合 MIPS 计算机进行浮点数加法的主要步 ，给出详细的计算步 ，结果不需要转换成 IEEE754 标准浮点数。（6 分）

2）IEEE754 标准中对单精度浮点数用 32 个 bits 来表示，其中最 位为浮点数的符号位，指数域为 8 位宽，尾数域 23 位宽，表示方式如下图所示：

| Bit 31 | Bit 30 ～ 23 | Bit 22～ 0 |
|---|---|---|
| Sign | 指数域 8 bits | 尾数域 23 bits |

请将 $1.000_2 \times 2^{-1}$ 表示为 IEEE754 标准中的单精度浮点数形式。（4 分）

2.用补码计算 X+Y 和 X-Y,并判定是否溢出及溢出的形式。其中 X=+1000, Y=+1001,数值位 4 位，采用双符号位。

答：X+Y = 01 0001　发生正上溢

　　X - Y = 11 1111　未溢出

3. 将十进制数 0.75 和 20.59375 转为 IEEE754 单精度浮点数的十进制形式。

4. IEEE754 单精度浮点数的二进制形式 0|10000001|01000...00 的十进制真值是多少，写出具体过程。

答：9.0　过程略

5. MIPS 计算机中 IEEE754 单精度浮点数表示十进制数-0.75 后得到的 16 进制结果是多少，写出具体过程。

## 3. 计算 Cache 组相联需要多少位，计算直接映射、二路组相联、全相联的命中率等

假设cache大小为$2^n$个块，块大小为$2^m$个字($2^{m+2}$个字节)

| valid | tag | index | offset |
|---|---|---|---|
| 1 | 32-n-2-m | n | 2+m |
| **说明：固定偏移两位是因为 mips 是按字节寻址，且大段对齐。比如进入一个地址 001101，计算机先自己扩展为 00110100，00 为偏移量，然后在此基础上由低位向高位进行截断。** | | | |

1. 直接映射：(块地址) mod (cache 中的块数)

总位数：$2^n \times (\text{块大小}[2^m \times 32] + tag + valid)$ 位

注意是什么单位：1 字=4 字节(B)=32 位

2. 计算命中率(以 4 个块为例，模板如下)

| 访问地址 | Miss/hit | 引用后 cache 的内容 | | | |
|---|---|---|---|---|---|
| 直接映射 | | 组 0 | 组 1 | 组 2 | 组 3 |
| 二路组相联 | | 组 0 | 组 0 | 组 1 | 组 1 |
| 全相联 | | 块 0 | 块 1 | 块 2 | 块 3 |

3. 练习题：

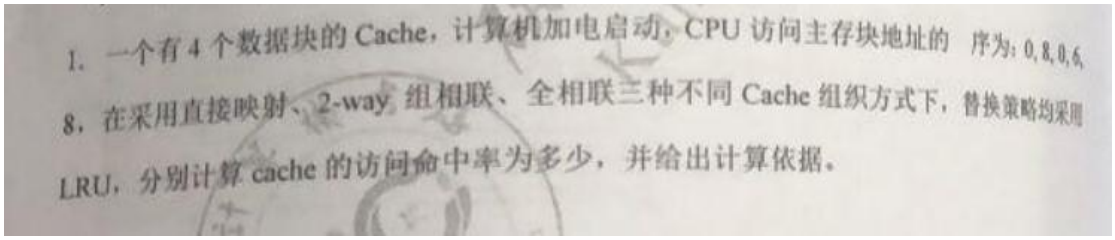
3.1 计算机的字长为 32 位，假设主存的最大容量为 8MB,Cache 中数据容量为 64KB,内存与 Cache 交换数据块的大小为 16 个字节，若按照采用直接映射方式。请问：

1)Cache 划分为多少块?每个块中包含多少个字

2)使用物理地址访问 Cache 时，物理地址的划分情况，并说明每个字段的位数及在物理地址中的位置。

3)请计算 Cache 总的容量有多大(需要考虑有效位和标记位).

3.2 回答题目并简述 cache 容量、块大小和相联度对 cache 性能的影响。




当其它设置都相同时,随着 cache 容量的增加,Cache 的命中率会增加,但是命中时间可能会上升；

当其它设置相同,随着增加块的大小时,Cache 命中率会先上升然后会下降,同时可能会增加未命中的代价。

当其它都相同时,增加相联性,会提高命中幸,但是代价是导致命中时间上升。

3.3 替换策略有哪些？适用范围是什么？

直接映射：直接替换，没有选择

LRU：最近最少使用，在二路或四路组相联中效果较好，更高相联度可能较差。

随机替换策略

## 4. Cache 性能评估：计算一二级 cache 下的实际 CPI

计算公式：AMAT(平均存储器访问时间) = 命中时间 + 缺失率×缺失代价

常见的单位：1GHz = $10^9$ Hz 对应 1ns

可以转化为 CPI，也可以直接计算时间：

**CPU Time = clock cycles × Cycle time = clock cycles/clock rate**

**= I(指令数)×CPI × Cycle time**

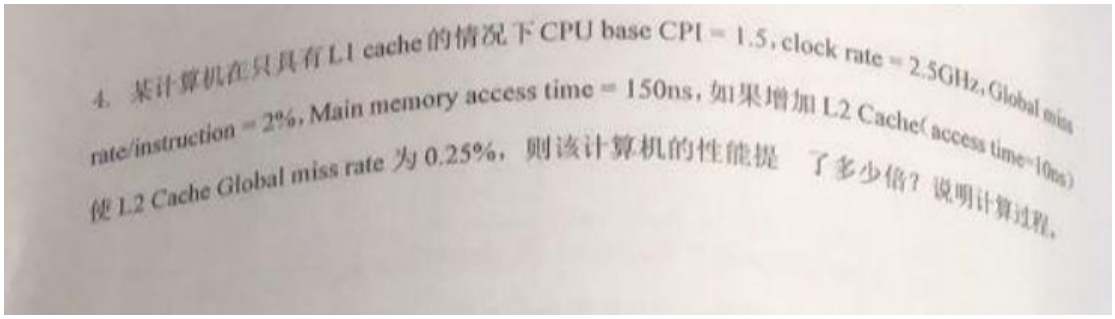
总公式

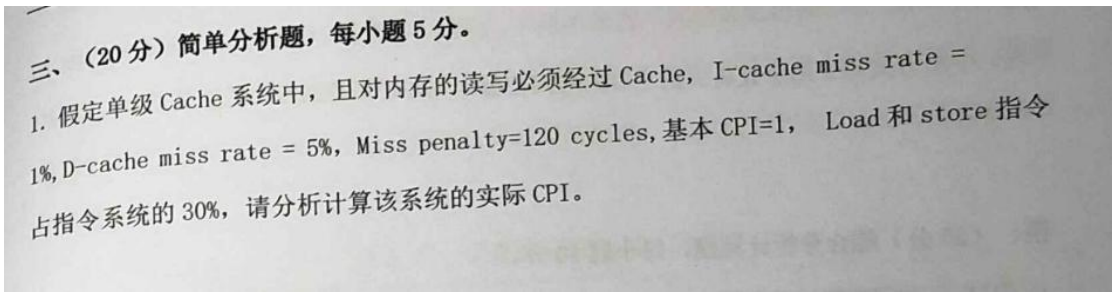
**一级 cache：实际 CPI = 基本 CPI + 一级缺失率×缺失周期数(=频率×访存时间)**

**二级 cache：实际 CPI = 基本 CPI + 一级缺失率×缺失周期数(=频率×二级 cache 访问时间)+二级缺失率×缺失周期数(=频率×访存时间)**

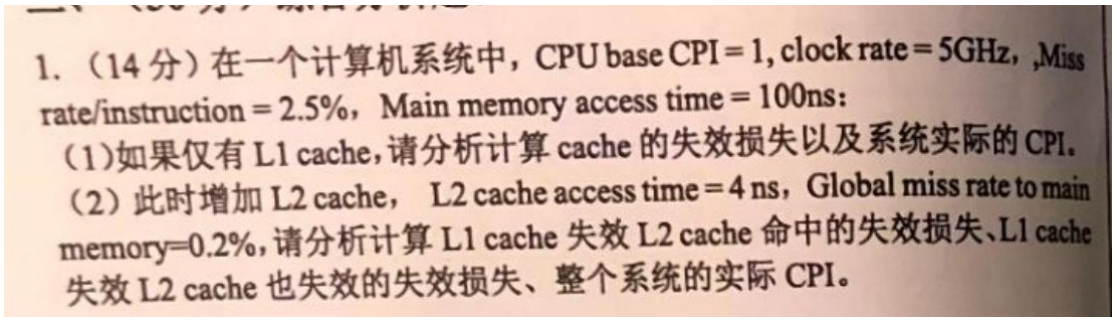
注意：如果分了 I-cache 和 D-cache 还要考虑(× load/store 比例) 根据题目来。

练习题：


4. 某计算机在只具有L1 cache的情况下CPU base CPI = 1.5, clock rate = 2.5GHz, Global miss rate/instruction = 2%, Main memory access time = 150ns, 如果增加L2 Cache(access time=10ns)使L2 Cache Global miss rate 为0.25%，则该计算机的性能提 了多少倍？说明计算过程。


三、（20分）简单分析题，每小题5分。

1. 假定单级Cache系统中，且对内存的读写必须经过Cache，I-cache miss rate = 1%, D-cache miss rate = 5%，Miss penalty=120 cycles, 基本CPI=1， Load和store指令占指令系统的30%，请分析计算该系统的实际CPI。


1.（14分）在一个计算机系统中，CPU base CPI = 1, clock rate = 5GHz，Miss rate/instruction = 2.5%，Main memory access time = 100ns：
（1）如果仅有L1 cache，请分析计算cache的失效损失以及系统实际的CPI。
（2）此时增加L2 cache， L2 cache access time = 4 ns，Global miss rate to main memory=0.2%，请分析计算L1 cache失效L2 cache命中的失效损失、L1 cache失效L2 cache也失效的失效损失、整个系统的实际CPI。

答：

(1)*cache*失效损失CPI：$2.5\% \times 5 \times 10^{9} \times 100 \times 10^{-9} = 12.5$

系统实际的CPI：1+12.5=13.5

(2)*L1cache*失效*L2cache*命中的失效损失为：$2.5\% \times 5 \times 10^{9} \times 4 \times 10^{-9} = 0.5$

*L1cache*失效*L2cache*也失效的失效损失为：$2.5\% \times 0.2\% \times 5 \times 10^{9} \times 100 \times 10^{-9} = 0.025$

整个系统实际的*CPI*为：$1 + 0.5 + 0.025 = 1.525$

## 5. 流水线性能分析

$$\text{Pipeline speedup} = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall cycles from branches}}$$

$$\text{Pipeline stall cycles from branches} = \text{Branch frequency} \times \text{Branch penalty}$$

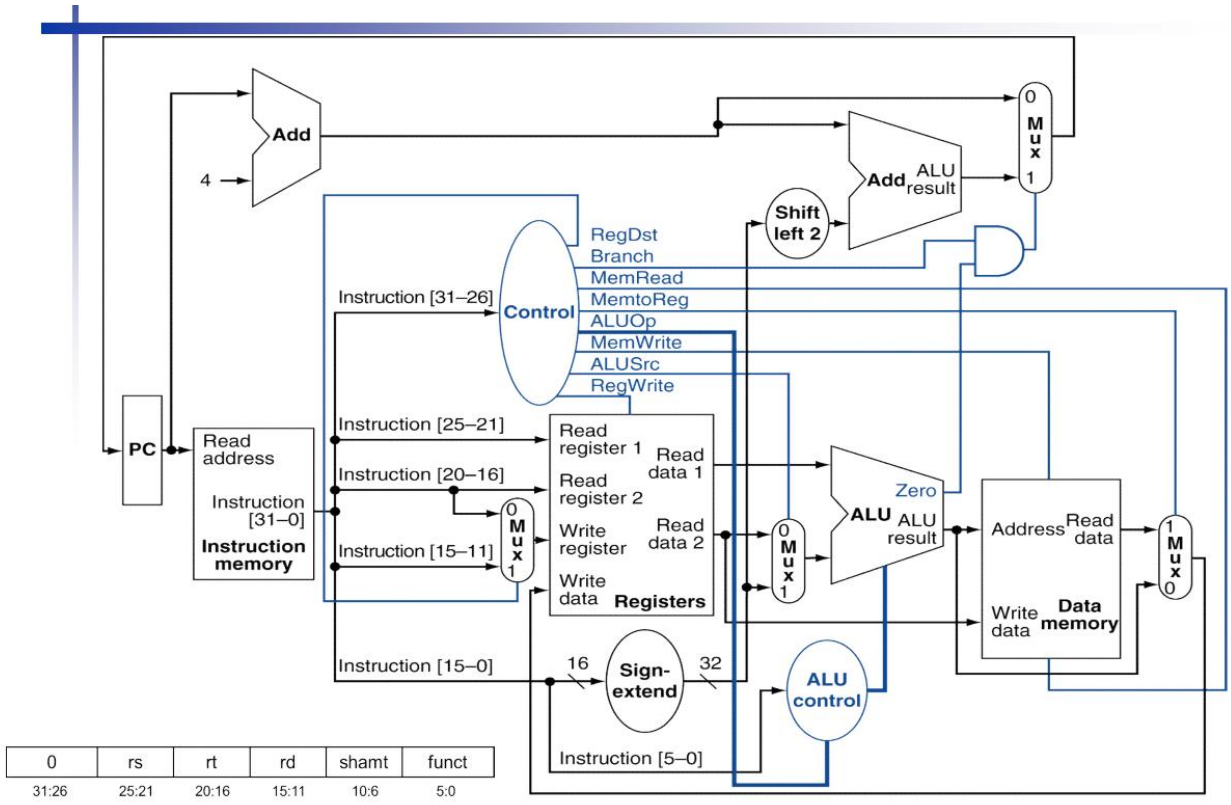
$$\text{Pipeline speedup} = \frac{\text{Pipeline depth}}{1 + \text{Branch frequency} \times \text{Branch penalty}}$$

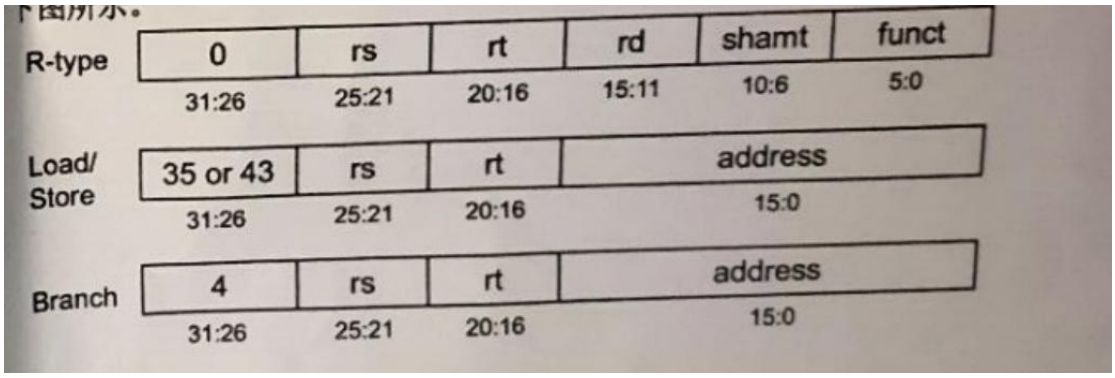
**6. 流水线数据冒险**

RAW,WAW,WAR

# 分析题

1. **分析 R 型指令，lw，sw，beq，jump 指令控制信号，关键路径和指令周期长度。**




信号表

| 指令 | Regdst | Alusrc | Memtoreg | Regwrite | Memread | Memwrite | Branch | Aluop | Jump |
|---|---|---|---|---|---|---|---|---|---|
| R 型 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 00 | 0 |
| lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 01 | 0 |
| sw | X | 1 | X | 0 | 0 | 1 | 0 | 01 | 0 |
| beq | X | 0 | X | 0 | 0 | 0 | 1 | 10 | 0 |
| jump | X | X | X | 0 | 0 | 0 | X | XX | 1 |
| Addi | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 00 | 0 |

aluop 根据题目来写

主要单元的延迟时间表：ps,其他单元忽略

| Imem | Adder | Mux | Alu | RF | Dmem | SignExt | CU |
|---|---|---|---|---|---|---|---|
| 400 | 100 | 30 | 120 | 200 | 350 | 20 | 100 |

关键路径及时间表

| 顺序 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 总时间 |
|---|---|---|---|---|---|---|---|---|
| R 型 | Imem | RF | Mux | Alu | Mux | RF | | 980ps |
| lw | Imem | RF | Mux | Alu | Dmem | Mux | RF | 1330ps |
| sw | Imem | RF | Mux | Alu | Dmem | | | 1000ps |
| beq | Imem | RF | Mux | Alu | Mux | | | 780ps |
| Jump | 根据图来决定路径 | | | | | | | |

描述路径的话，以下给出举例：

(一)关于加法指令

加法执行过程如下：

(1)PC 寄存器中地址送指令存储器，并读指令；PC 地址同时送地址加法部件实现 PC+4;

(2)根据指令中的寄存器的 rs 和 rt 的编号从寄存器堆中读出两个寄存器中的值；

(3) rs 和 rt 中的值送入运算器中进行加法运算；

(4) 运算的结果送入寄存器堆中的 rd 寄存器；

(5)将 PC+4 的值送入 PC

该指令执行的关键路径为：

取指令->取操作数(同时控制器译码)->多路选择器一运算器>多路选择器->写寄存器堆

因此总的时间为：400ps+200ps+30ps+120ps+30ps+200ps=980ps

(二)关于跳转指令执行的时间

(1)PC 寄存器中地址送指令存储器，并读指令；PC 地址同时送地址加法部件；

(2)根据指令中的寄存器的 rs 和 rt 的编号从寄存器堆中读出两个寄存器中的值，同时将指令中低/16 位送入带符号位的扩展电路，并将将扩展后的数据送左移部件后进行与 PC+4 的值进行加法操作；

(3) rs 和 rt 中的值送入运算器中进行比较(减法)运算；

(4)根据运算结果，选择 PC+4 或者 PC+4+16 位偏移地址送入 PC;该指令执行的关键路径为：

取指令->取操作数(同时控制器译码)->多路选择器->运算器->多路选择器(根据结果选择地址送 PC)

因此总的时间为：400ps+200ps+30ps+120ps+30ps=780ps

**2. 将 C 语言转化为 mips 指令，看书，注意 J，jal，ra，sp(栈的使用)，循环 Loop 的使用**

**书上原题！自己找**

2. 请给出下列 C 语言代码对应的 MIPS-32 汇编语言代码。

```
void strcpy (char x[], char y[])
{ int i;
  i = 0;
  while ((x[i]=y[i])!='\0')
    i += 1;
}
```

Addresses of x, y in $a0, $a1，i in $s0。

## 3. Branch 分支预测性能比较图：理解+识记

### Example: Branch Penalty (in CPI units) on the MIPS R4000 Processor

| Branch scheme | Penalty for jumps | Penalty for untaken branches | Penalty for taken branches |
|---|---|---|---|
| Flush Pipeline | 2 | 3 | 3 |
| Predict Taken | 2 | 3 | 2 |
| Predict Untaken | 2 | 0 | 3 |

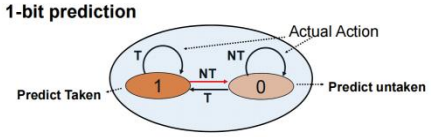
MIPS R4000 takes 3 cycles to compute the target address (i.e. minimum 2 CPI of *additional* penalty) and 4 cycles to know the branch outcome

| Strategy | Jumps | Untaken branches | Taken branches | Combined |
|---|---|---|---|---|
| Frequency (est.) | 4% | 6% | 10% | 20% |
| Flush Pipeline | 0.08 | 0.18 | 0.30 | 0.56 |
| Predict Taken | 0.08 | 0.18 | 0.20 | 0.46 |
| Predict Untaken | 0.08 | 0.00 | 0.30 | 0.38 |

●5

## 4. Branch 分支预测：

### 4.1 局部：1 位，2 位状态转移图以及会判断分析




### Example: 2-Bit Prediction Scheme in Action with "Loop Branching"

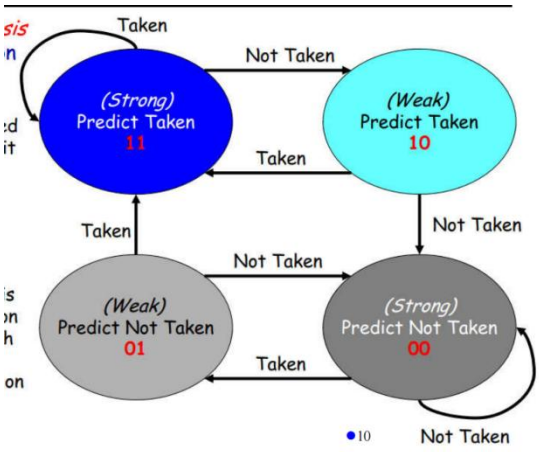
```
loop: L.D F0, 0(R1)
      MUL.D F4, F0, F2
      S.D F4, 0(R1)
      DADDIU R1, R1, #-8
      BNEZ R1 loop
```

- Assumptions
  - R1 is initialized to #80

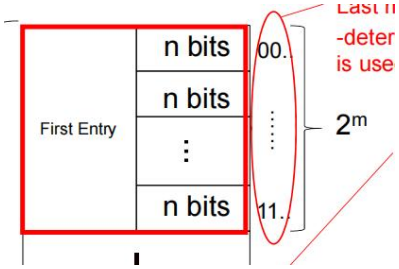
| Iterationss & steps | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Predicted behavior | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T |
| Actual behavior | T | T | T | T | T | T | T | T | T | T | NT | T | T | T | T | T | T | T | T | T | T | NT |

- Branch taken 90% of the times, and branch prediction accuracy is now 90%
- The 2-bit predictor mispredicts at the 10th step of the 1st iteration, but *doesn't change its mind*. It just moves to "weak-predict-taken" for the 1st step of the following iteration

●11

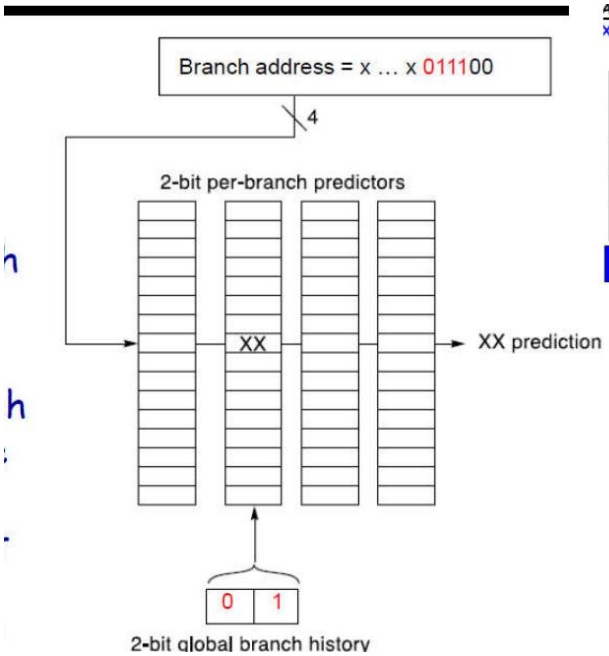



**4.2 全局 (m,n)级预测器，最近 m 个分支，用 n 位进行预测，结构图：**




以(1,1)举例，右图， if d = =0 then d = 1;

If d = =1 then .....;




Assumption: d alternates between 2 and 0
X/Y: use X if last branch was not taken,
use Y if last branch was taken

L2 : …

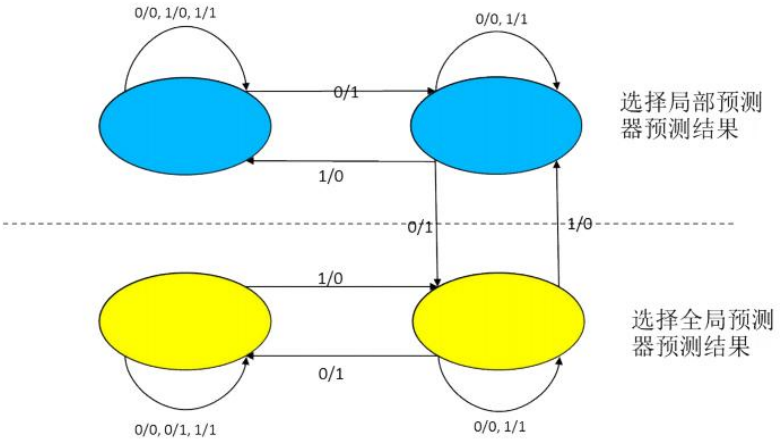
| d=? | b1 pred. | b1 action | b1 new pred. | b2 pred. | b2 action | b2 new pred. |
|---|---|---|---|---|---|---|
| 2 | NT/NT | T | T/NT | NT/NT | T | NT/T |
| 0 | T/NT | NT | T/NT | NT/T | NT | NT/T |
| 2 | T/NT | T | T/NT | NT/T | T | NT/T |
| 0 | T/NT | NT | T/NT | NT/T | NT | NT/T |

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

简要解释一下(m,n)分支历史记录：以(3,2)为例，假设分支历史真实值为.....TTTNNTNT, n=2 代表有限状态机有 $2^n$ 个状态，见 4.1 的最后一张图，然后共有 $2^m$=8 个这样的状态机(编号为 000，001，010.....，111)，都可以用于预测。具体选择哪个，选择最近 m 条历史记录对应的编号(本例中是 TNT,即 101 编号的状态机)对应的状态机进行预测，再根据真实值进行状态转换，以此类推。

**4.3 竞赛预测器：结合局部＋全局：用饱和计数器，原理简单来说就是用局部 A 和全局 B 预测器，用一个计数器 Cnt 计数。假如当前用 A 进行预测，当 A 和 B 都预测正确或**

**都错误，保持当前 A 进行预测;当 A 正确 B 错误，保持 A；当 A 错误一次，B 正确，转入 A 的过渡态----在此基础上，A 错误 B 正确，则转入 B 进行预测。也就是 A 连续错误两次且 B 连续正确两次，控制权由 A 交给 B。状态转移图如下：**




例题：

1. A snapshot of the taken/not-taken behavior of a branch is: ... T T T T T T T T N N T T N N T N N T

If the branch predictor used is a 2-bit saturating counter, how many of the last ten branches are predicted correctly?

Answer：

According to the 2-bit branch predictor, the prediction for the last ten branches are:

ST, WT, SN, WN, ST, WT, SN, WN, SN, SN (S for strong, W for weak)

N N T T N N T N N T

According to the 2-bit saturating counter:

ST, WT, WN, WT, ST, WT, WN, WT, WN, SN

N N T T N N T N N T

No matter which one you use, the answer is 2 branches are predicted correctly.

2. Assume a machine that has a branch-target buffer with 8 entries. A branch in this machine has a penalty of 2 clock cycles if the branch is taken and the target instruction is not in the branch-target buffer, or if the branch is predicted as taken, the instruction is in the branch-target buffer, but the branch is actually not taken. In all other situations the branch penalty is zero. What is the total branch penalty in this machine, measured in clock cycles, if

the branch prediction accuracy is 90%;

80% of the time the target instruction is in the buffer (80% hit rate in the buffer);

60% of the branches are actually taken.

Answer：

Probability of branch taken but not found in the buffer:

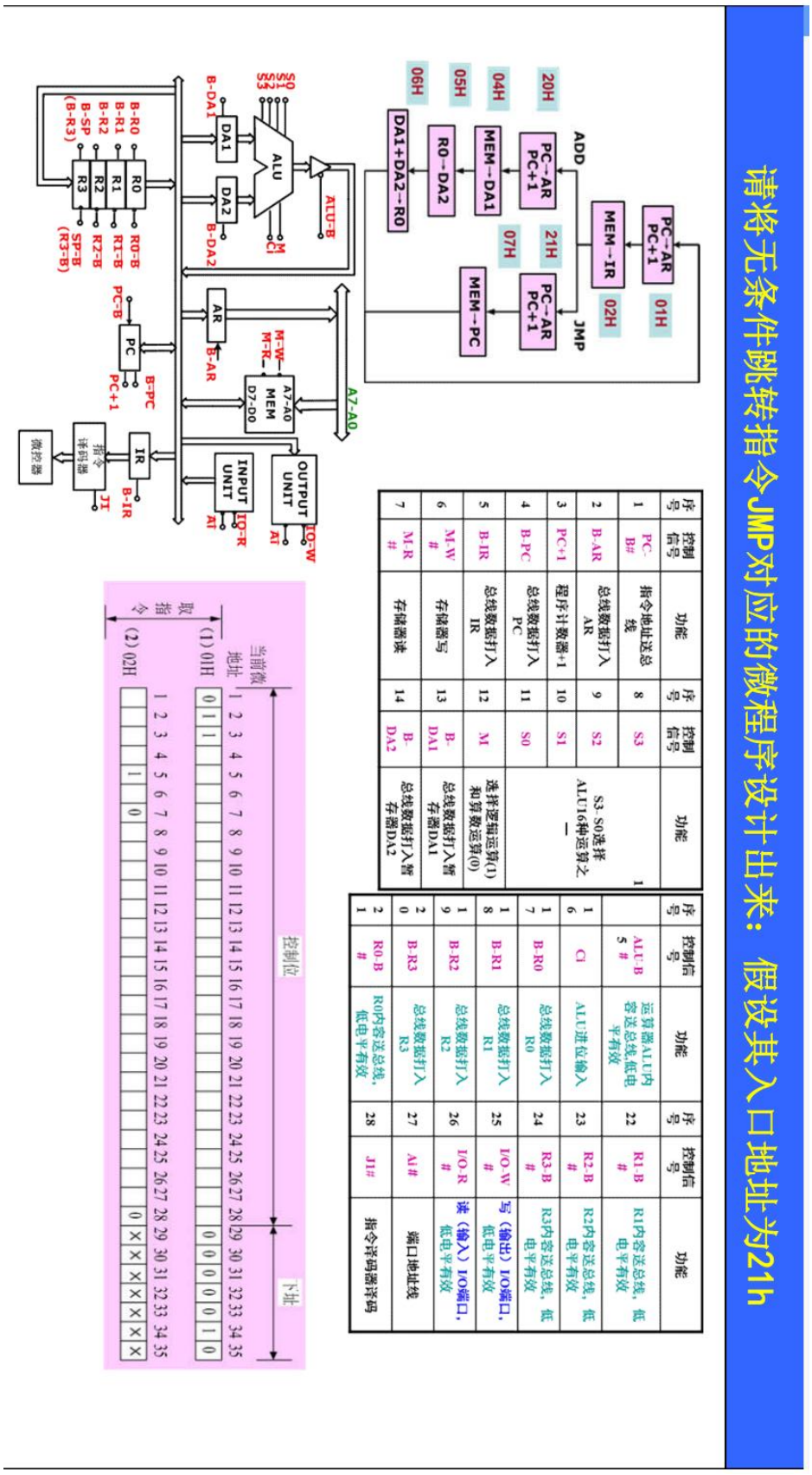
Percentage of taken branches * buffer miss rate = 60% * 20% = 0.12

Probability of branch found in buffer but predicted wrong:

Buffer hit rate * prediction miss rate = 80% * 10% = 0.08

Average branch penalty = ( 0.12 + 0.08 ) * 2 = 0.4 clock cycles

~~5. 微程序设计(自求多福啦！)核心图：~~




| 序号 | 控制信号 | 功能 | 序号 | 控制信号 | 功能 |
|---|---|---|---|---|---|
| 1 | PC B# | 指令地址送总线 | 8 | S3 | S3-S0选择ALU16种运算之一 |
| 2 | B-AR | 总线数据打入AR | 9 | S2 | |
| 3 | PC+1 | 程序计数器+1 | 10 | S1 | |
| 4 | B-PC | 总线数据打入PC | 11 | S0 | |
| 5 | B-IR | 总线数据打入IR | 12 | M | 选择逻辑运算(1)和算数运算(0) |
| 6 | M-W # | 存储器写 | 13 | B-DA1 | 总线数据打入暂存器DA1 |
| 7 | M-R # | 存储器读 | 14 | B-DA2 | 总线数据打入暂存器DA2 |

| 序号 | 控制信号 | 功能 | 序号 | 控制信号 | 功能 |
|---|---|---|---|---|---|
| 16 | ALU-B # | 运算器ALU内容送总线,低电平有效 | 22 | R1-B # | R1内容送总线,低电平有效 |
| 17 | Ci | ALU进位输入 | 23 | R2-B # | R2内容送总线,低电平有效 |
| 18 | B-R0 | 总线数据打入R0 | 24 | R3-B # | R3内容送总线,低电平有效 |
| 19 | B-R1 | 总线数据打入R1 | 25 | I/O-W # | 写(输出)IO端口,低电平有效 |
| 20 | B-R2 | 总线数据打入R2 | 26 | I/O-R # | 读(输入)IO端口,低电平有效 |
| 21 | B-R3 | 总线数据打入R3 | 27 | AI # | 端口地址线 |
| 22 | R0-B # | R0内容送总线,低电平有效 | 28 | JI # | 指令译码器译码 |

~~6. 总线：我不会呜呜呜~~

## 7. 指令寻址方式

立即数寻址、寄存器寻址、基址寻址、PC 相对寻址、伪直接寻址的指令格式和方式。

寄存器寻址：

操作数是寄存器

基址寻址：

操作数在内存中，其地址是指令中的基址寄存器和常数的和

立即数寻址：

操作数是指令自身中常数

PC 相对寻址：

16 位偏移地址左移两位+PC

伪直接寻址：

26 位偏移地址+PC 高 4 位　30 位字地址

# 简答题

### 1. 简述虚拟存储器的工作原理？其对现代计算机存储层次的支持作用何在？

**工作原理：**虚拟存储器是一种将主存用作辅助存储器的高速缓存技术，将主存划分为固定大小的块(页面),并将其映射到辅助存储器上的存储区域产生虚拟地址，当程序访问一个页时，虚拟存储器结合软硬件转化为物理地址，用来访问主存。

**支持作用：**

**①扩展内存容量：**虚拟存储器允许程序使用**比物理内存更大**的地址空间。

**②内存管理：**一种灵活的方式来管理内存资源。

**③内存保护：**虚拟存储器系统通过**使用页表和访问权限位**来实现内存保护。

**④隔离和安全性：**虚拟存储器系统为每个运行的程序提供了**独立的地址空间**，使它们彼此隔离。

### 2. 计算机五大经典部件及作用？

**—输入：**键盘、鼠标等数据的输入

**—控制器：**负责控制所有硬件的运行

**—数据通路(运算器)：**实现算术运算和逻辑运算

**—存储器：** 内存：易失性存储器，由 DRAM 组成，程序运行时用来存储数据的空间

（cache 是使用 SRAM，速度更快，更小）

外存：非易失性存储器，例如硬盘，能永久保存数据

**—输出：**显示器、打印机等数据输出

CPU(中央处理单元) = 控制器 + 数据通路

### 3. 简要分析算法、编程语言和编译器、指令系统对计算机系统性能的影响。

①算法决定源程序指令的数量及程序的 CPI；

②编程语言主要影响指令的条数和 CPI；

③编译器主要影响指令的条数和 CPI；
④指令系统决定指令数、时钟频率和 CPI.
补：程序语言执行的过程：高级语言程序→汇编语言程序→机器语言程序
扩：处理器和内存决定指令执行的速度快慢，I/O 决定操作执行的速度快慢

**4. 简述性能的定义和度量方式。**

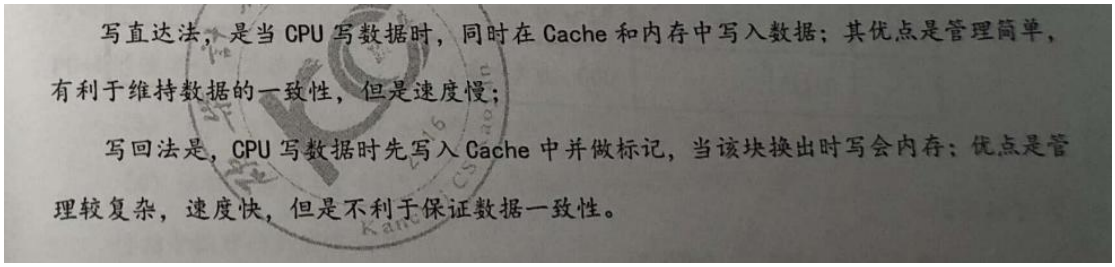
①性能可以由执行时间和吞吐率(带宽)描述：

执行时间：计算机完成某任务的总时间，包括 I/O 活动，CPU 执行时间，内存访问，操作系统开销等。

吞吐率：也叫带宽，单位时间内完成的任务数量。

②用 CPU 执行时间度量：包括用户 CPU 时间(程序自身)和系统 CPU 时间(操作系统)。

**5. 简述 cache 写直达和写回机制的概念和优缺点。**


写直达法，是当 CPU 写数据时，同时在 Cache 和内存中写入数据；其优点是管理简单，有利于维持数据的一致性，但是速度慢；

写回法是，CPU 写数据时先写入 Cache 中并做标记，当该块换出时写会内存；优点是管理较复杂，速度快，但是不利于保证数据一致性。

**6. 流水线中可能会出现哪些冒险？阐述其定义及可能解决方法有哪些？**

(1) 结构冒险：由缺乏硬件支持而导致指令无法在预定的时钟周期内完成。

解决方法：由于增加流水线硬件单元代价较高，若一个结构冒险很罕见，则可以不用去控制。

(2) 数据冒险：由于无法提供指令执行所需的数据而导致指令无法在预定的时钟周期内完成。

解决方法：插入 nop 指令延迟、重排指令编译时间顺序、数据前推或旁路。

(3) 控制冒险：由于取到的指令不是所需要的而导致指令无法在预定的时钟周期内完成。

解决方法：采用分支预测技术，如静态分支预测、动态分支预测、竞赛器分支预测等

**7. 简述 cache 中常见的三种缺失的概念及解决措施。**

(1) 强制缺失：由于在 cache 中缺少相应的块第一次进行访问而引起的缺失。

解决方法：增加块的大小---增加损失代价。

(2) 容量缺失：cache 本身无法容纳所有请求的块。

解决方法：增加 cache 的容量

(3) 冲突缺失：很多块竞争同一个组而引起的缺失

解决方法：增加相联度、增加 cache 的大小

**8. 简述虚拟存储器 TLB 的工作机制，列举出至少三种减少页表储存的主存存储量的措施。**

**9. 简述单重中断系统的处理过程。**

关中断、保存现场、判断中断源进入中断服务程序，开中断，中断处理程序，关中断，恢复现场，返回主程序。

**10. 冯·诺依曼型计算机的基本特点是什么？**

• 采用二进制形式表示数据和指令。指令由操作码和地址码组成。

• 将程序和数据存放在存储器中，使计算机在工作时从存储器取出指令加以执行，自动完成计算任务。这就是“存储程序”和“程序控制”（简称存储程序控制）的概念。

• 指令的执行是顺序的，即一般按照指令在存储器中存放的顺序执行，程序分支由转移 指令实现。 • 计算机由存储器、运算器、控制器、输入设备和输出设备五大基本部件组成，并规定了 5 部分的基本功能。

冯•诺依曼型计算机的基本特点也可以用“存储程序”和“程序控制”来高度概括。

**11. Cache 有哪些特点？**

(1) 位于 CPU 与主存之间，是存储器层次结构中级别最高的一级。

(2) 容量比主存小，目前一般有数 KB 到数 MB。

(3) 速度一般比主存快 5~10 倍，通常由存储速度高的双极型三极管或 SRAM 组成。

(4) 其容量是主存的部分副本。

(5) 可用来存放指令，也可用来存放数据。

(6) 快存的功能全部由硬件实现，并对程序员透明。

**12. 什么是中断？中断技术给计算机系统带来了什么作用？**

当计算机执行正常程序时，系统中出现某些异常情况或特殊请求，CPU 暂停它正在执行的程序，而转去处理所发生的事件；CPU 处理完毕后，自动返回到原来被中断了的程序继续运行。

中断的作用：（1）主机与外部设备并行工作；（2） 实现实时处理；（3）硬件故障处理；（4）实现多道程序和分时操作。

***Version1:***

***写在最后，编者水平能力有限，望各位大佬指正和补充！***

***2023.6.17***

***在图书馆挣扎的 mzq***

***编者不易，请勿随意发布该文档。***

***Version2:***

***2023.6.18***

***• 修改了部分错误：修正 2.1 操作数 y 的补码，修正分析题 1 的 beq-jump 信号，以及其他描述错误。***

***• 增加(m,n)全局预测器解释、竞赛预测器原理说明***

***• 完善简答题 6，7 题***