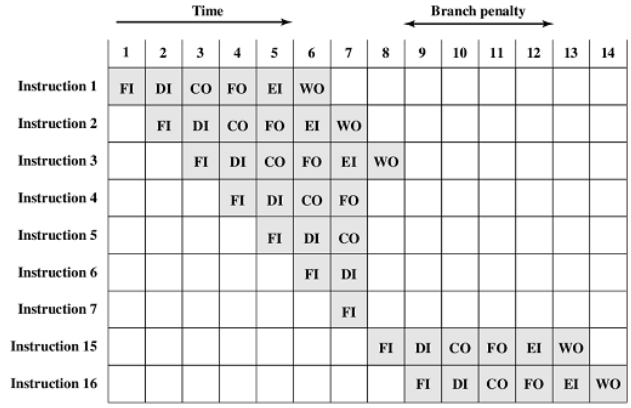
“计算机组织结构”作业 09 参考答案

1. 考虑一个通过指令流水线来处理的长度为 n 的指令序列。假设遇到一条有条件或无条件转移指令的概率为 p，并假设执行转移 I 时转移到非连续地址的概率是 q。请重新写出使用 k 段流水线执行 n 条指令所需总时间的公式和加速比公式。

（为简化问题，认为只当发生转移的指令 I 在流水线上最后一段刚一出现时，总清流水线并撤销线上正在进行的指令。）



总时间$T = [k + (n - 1)]t + pqn(k - 2)t$

加速比$S_k = \frac{nkt}{[k+(n-1)]t+pqn(k-2)t} = \frac{nk}{k+n-1+pqn(k-2)}$

2.一时钟速率为 2.5GHz 的流水式处理器执行一个有 1.5 百万条指令的程序。流水线有 5 段并以每时钟周期 1 条的速率发射指令。不考虑转移指令和无序执行所带来的性能损失。

(1)同样执行这个程序，该处理器比非流水式处理器加速了多少（百分数）？

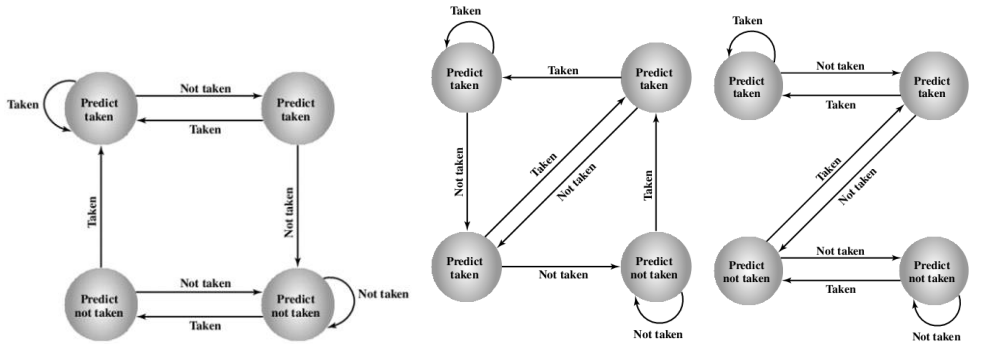
(2)此流水式处理器的吞吐率是多少（以 MIPS 为单位）？

(1)加速比 S_k=nkt/([k+(n-1)]t)=k/(1+(k-1)/n)

由于有 1.5 百万条指令，即 n 很大，所以 S_k 为 k，即 5，加速了 **400%**。

(2)由于近似于每个周期完成一条指令，所以吞吐率为 2.5G/106=**2500** MIPS。

3.假设使用下面 3 种转移处理状态图 A、B、C



（A） （B） （C）

执行以下一段程序

```
int sum (int N) {
        int i, j, sum = 0;
for (i = 0; i < N; i++)
    for (j = 0; j < N; j++)
        sum = sum + 1;
return sum;
}
```

相应的汇编程序段为

```
        …
Loop-i: beq $t1, $a0, exit-i          # 若（i=N）则跳出外循环
        add $t2, $zero, $zero          # j=0
Loop-j: beq $t2, $a0, exit-j          # 若（j=N）则跳出内循环
        addi $t2, $t2, 1               # j=j+1
        addi $t0, $t0, 1               # sum=sum+1
        j Loop-j
exit-j: addi $t1, $t1, 1          # i=i+1
        j Loop-i
exit-i: …
```

假设算法从流程图的左上角开始：

a) 分析 N=10 时，使用转移处理状态图 A 的外层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

b) 分析 N=10 时，使用转移处理状态图 A 的内层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

c) 分析 N=100 时，使用转移处理状态图 A 的外层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

d) 分析 N=100 时，使用转移处理状态图 A 的内层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

e) 分析 N=10 时，使用转移处理状态图 B 的外层 for 循环预测正确率（百分数，

精度：小数点后 2 位）。

f) 分析 N=10 时，使用转移处理状态图 B 的内层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

g) 分析 N=100 时，使用转移处理状态图 B 的外层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

h) 分析 N=100 时，使用转移处理状态图 B 的内层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

i) 分析 N=10 时，使用转移处理状态图 C 的外层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

j) 分析 N=10 时，使用转移处理状态图 C 的内层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

k) 分析 N=100 时，使用转移处理状态图 C 的外层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

l) 分析 N=100 时，使用转移处理状态图 C 的内层 for 循环预测正确率（百分数，精度：小数点后 2 位）。

外循环共预测 N+1 次，内循环共预测 N×(N+1)次。外循环和内循环各有一组预测位。

使用转移处理状态图 A 时：

预测初始位为 11，外循环中第 1 次、第 2 次和最后一次预测错误，共错误 3 次。内循环中第 1 次进入时变成 10（不发生，预测错误），然后变成 00（不发生，预测错误），跳出时又变成 10（发生，预测错误）；其后每次进入时变成 00（不发生，预测正确），跳出时又变成 01（发生，预测错误），所以内循环共有 N+2 次预测错误。

a) N=10：外循环正确率　1-3/11=72.73%
b) N=10：内循环正确率　1-12/110=89.09%
c) N=100：外循环正确率　1-3/101=97.03%
d) N=100：内循环正确率　1-102/10100=98.99%

使用转移处理状态图 B 时：

预测初始位为 11，外循环中第 1 次、第 2 次和最后一次预测错误，共错误 3 次。内循环中第 1 次进入时变成 01（不发生，预测错误），然后变成 00（不发生，预测错误），跳出时又变成 10（发生，预测错误）；其后每次进入时变成 01（不发生，预测错误），然后变成 00（不发生，预测错误），跳出时又变成 10（发生，预测错误），所以内循环共有 3N 次预测错误。

e) N=10：外循环正确率　1-3/11=72.73%
f) N=10：内循环正确率　1-30/110=72.73%
g) N=100：外循环正确率　1-3/101=97.03%
h) N=10：内循环正确率　1-300/10100=97.03%

使用转移处理状态图 C 时：

预测初始位为 11，外循环中第 1 次、第 2 次和最后一次预测错误，共错误 3 次。内循环中第 1 次进入时变成 10（不发生，预测错误），然后变成 01（不发生，预测错误），然后变成 10（不发生，预测正确），跳出时又变成 01（发生，预测错误）；其后每次进入时变成 00（不发生，预测正确），跳出时又变成 01（发生，预测错误），所以内循环共有 N+2 次预测错误。

i) N=10：外循环正确率　1-3/11=72.73%
j) N=10：内循环正确率　1-12/110=89.09%
k) N=100：外循环正确率　1-3/101=97.03%
l) N=100：内循环正确率　1-102/10100=98.99%
[伍佳艺，141250150]
注：中间那张处理状态图的上面两个状态实际上可以合并[朱宇翔，141250216]

4. 如图 1 所示，假设沿总线和通过 ALU 的传播延迟分别为 20ns 和 100ns。由总线将数据拷贝到寄存器需要 10ns。

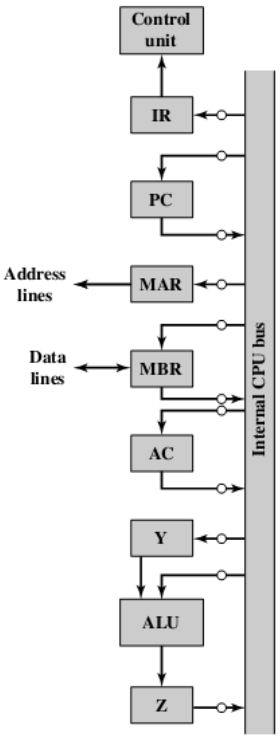


图 1

请问以下操作需要的最少时间为多少?

a) 将数据从一个寄存器传送到另一个寄存器;

在总线上传送数据用时 20ns，拷贝到寄存器中需要 10ns，总共需要 30ns

b) 使用 ALU 增量程序计数器。

从 PC 中读取地址传送到总线用时 20ns，放入 Y 中用时 10ns；经 ALU 中计算用时 100ns;将数据传输回 PC 用时 20ns+10=30ns;所以总共用时 160ns。

5. 控制器如图 2 所示。假定它的控制存储器是 24 位宽。微指令格式的控制部分分成两个字段。一个 13 位的微操作字段用来指定将要完成的微操作。一个地址选择字段用来指明能引起微指令转移的条件，这些条件是基于 8 个标志来建立的。

a) 地址选择字段有多少位?
b) 地址字段有多少位?
c) 控制存储器容量为多少（单位：字节）?

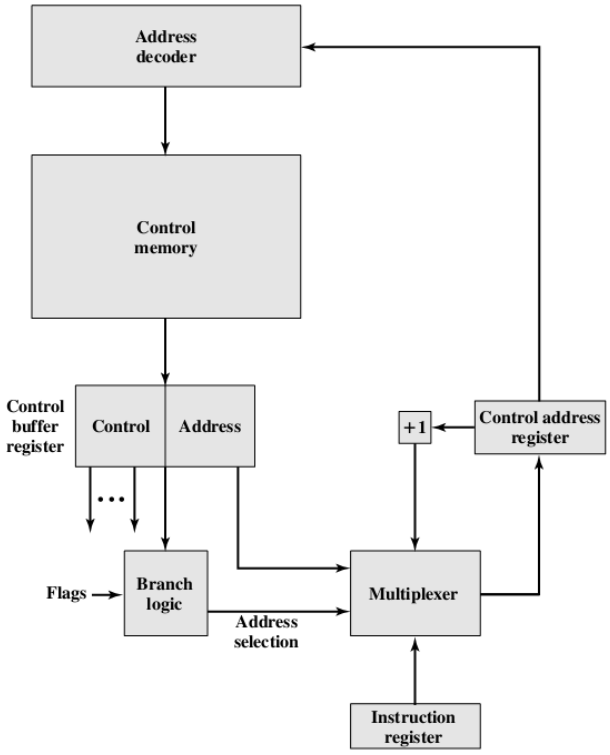


图 2

a) 地址选择字段共有 8 个标志，$\log_2 8=3$，故需要 3 位

b) 地址字段位数为 24-13-3=8 位

c) 地址为 8 位，表示微指令的条数最大为 2^8 =256 条，所以控制存储器的容量为 256 * 24/8 = 768 字节

6. 有一个 ALU 不能做减法，但它能加两个输入寄存器并能对两个寄存器的各位取逻辑反。其中，数据以二进制补码形式存储。请根据以下 4 种情形，列出用该 ALU 实现减法时控制器必须完成的操作。

a) 1 地址直接寻址

b) 1 地址间接寻址

c) 2 地址直接寻址

d) 2 地址间接寻址

假设 ALU 的两个输入寄存器为 X 和 Y。取反操作为 Neg()，加法为 Add()

a) 1 地址，直接寻址

t1: MAR <- (IR(address))

t2: MBR <- Memory

t3: Y <- (MBR)

t4: Y <- Neg(Y)

t5: X <- 1

t6: Y <- (X) + (Y)

t7: X <- (AC)

t8: AC <- (X) + (Y)

b) 1 地址，间接寻址

t1: MAR <- (IR(address))

t2: MBR <- Memory

t3: MAR <- (MBR)

t4: MBR <- Memory

t5: Y <- (MBR)

t6: Y <- Neg(Y)

t7: X <- 1

t8: Y <- (X) + (Y)

t9: X <- (AC)

t10: AC <- (X) + (Y)

c) 2 地址，直接寻址

t1: MAR <- (IR(address2))

t2: MBR <- Memory

t3: Y <- (MBR)

t4: Y <- Neg(Y)

t5: X <- 1

t6: Y <- (X) + (Y)

t7: MAR <- (IR(address1))

t8: MBR <- Memory

t9: X <- (MBR)

t10: AC <- (X) + (Y)

d) 2 地址，间接寻址

t1: MAR <- (IR(address2))

t2: MBR <- Memory

t3: MAR <- (MBR)

t4: MBR <- Memory

t5: Y <- (MBR)

t6: Y <- Neg(Y)

t7: X <- 1

t8: Y <- (X) + (Y)

t9: MAR <- (IR(address1))

t10: MBR <- Memory

t11: MAR <- (MBR)

t12: MBR <- Memory

t13: X <- (MBR)

t14: AC <- (X) + (Y)

7. 图 3 所示的栈保存在内存中，寄存器中存储了栈限（分配给该栈的最小地址）、栈指针（栈顶地址）和栈基（分配给该栈的最大地址）。请写出 push 和 pop 该栈所对应的微操作序列。

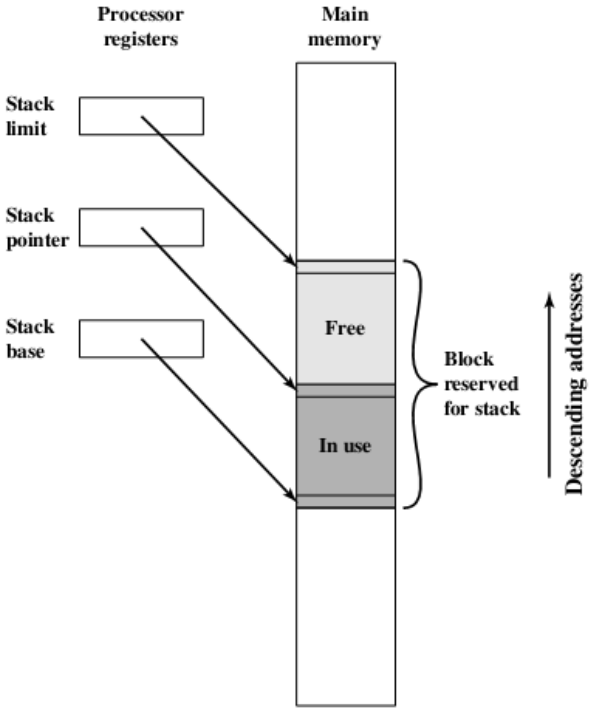


图 3

POP: t1: MAR <- (SP)

t2: MBR <- Memory

SP <- (SP) + 1

PUSH: t1: SP <- (SP) – 1

t2: MAR <- (SP)

t3: Memory <- (MBR)

8. 一个指令周期有 4 个主要阶段：取指、间址、执行和中断。硬布线方式实现时，采用一个 2 位的寄存器来标志当前阶段，但微程序式控制器却不需要类似的标志。请问为什么硬布线式控制器需要这些标志，而微程序式控制器不需要这些标志？

在硬连线式控制器中，当前阶段会作为输入的一部分，用于布尔逻辑式的计算，因此需要采用 2 位的寄存器来标志当前阶段。

在微程序式控制器中，所有的微指令都存储在控制存储器中，排序逻辑会确定下一条将要执行的微指令，各个阶段之间通过跳转来实现，因此不需要状态标志。

9. CPU 有 16 个寄存器，一个 ALU 有 16 种逻辑功能和 16 种算术功能，一个移位器有 8 种操作，所有这些组件都与一个 CPU 内部总线相连。假设 ALU 的输入和输出都位于寄存器中，设计一种微指令格式能指定此 CPU 的各种微操作。

微指令格式如下：

逻辑和算术功能（0～4位）	移位操作（5～7位）	ALU 输入1（8～11位）	ALU 输入2（12～15）位	ALU 输出（16-19位）