作业题（1）：3、4、5、6、7、8、10、11
作业题（2）：14、16、17、18、19、20、21、22、23

3．图 5.67 给出了某 CPU 内部结构的一部分，MAR 和 MDR 直接连到存储器总线（图中省略）。在两个总线之间的所有数据传送都需经过算术逻辑部件 ALU。ALU 的部分控制信号及其功能如下。

MOVa：F=A；　MOVb：F=B；
a+1：F=A+1；　b+1：F=B+1；
a-1：F=A–1；　b-1：F=B–1。

其中，A和B是ALU的输入，F是ALU的输出。假定该CPU对应指令系统中调用指令CALL占两个字，第一个字是操作码，第二个字给出子程序的起始地址，返回地址保存在栈中，用SP（栈指示器）指向栈顶，存储器按字编址，每次按同步方式从主存读取一个字，要求：

（1）说明CALL指令的功能。

（2）写出读取并执行CALL指令所要求的控制信号序列（提示：当前指令地址已在PC中），并说明至少需要多少个时钟周期。

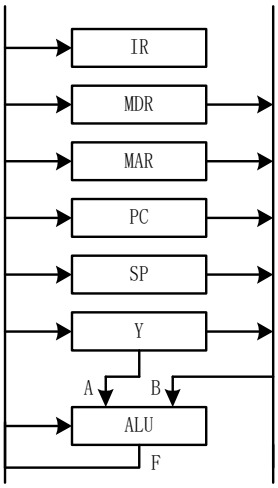


图 5.67　题 3 中的图示

【分析解答】

（1）CALL 指令的功能是：将返回地址（CALL 指令下条指令地址）压栈，然后跳转到目标地址处（子程序起始地址），即修改 PC 的值为目标地址。

（2）采用同步方式读写内存，即在 read 和 write 信号后不需加等待信号 WMFC。CALL 指令有两个字，按字编址，每次从主存读取一个字，因此，CALL 指令需要读两次主存，一次是读取指令中的操作码，另一次是读取指令中给出的子程序首地址。其指令周期分为以下三个阶段。

1）读取指令操作码：将 PC 的内容作为地址访问存储器，取出指令的操作码，送指令寄存器 IR，同时 PC+1 送 PC，以指向指令的第二个字，至少需要三个时钟周期（节拍）。

PC_{out}, MOVb, MAR_{in}
Read, b+1, PC_{in}
MDR_{out}, MOVb, IR_{in}

2）取子程序首地址：将 PC 的内容作为地址，取出指令的第二个字（即子程序入口地址）送 PC，以使下一个指令周期从子程序的第一条指令开始执行。同时，计算 PC+1 以得到返回地址，送 Y 寄存器，至少需要三个时钟周期。

PC_{out}, MOVb, MAR_{in}
Read, b+1, Y_{in}
MDR_{out}, MOVb, PC_{in}

3）保存返回地址至栈中：将临时保存在 Y 寄存器的返回地址送到栈顶保存，并自动调整栈顶指针。至少需要三个时钟周期。

SP_{out}, MOVb, MAR_{in}

Y_{out}, MOVb, MDR_{in}

Write, SP_{out}, b-1, SP_{in}

显然，上述每个节拍中执行的操作所需要时间不等，其中，存储访问（read/write）时间最长，时钟周期以最长的存储访问时间为准，CALL 指令的指令周期至少有 9 个时钟周期（节拍）。

如果将第一次 PC+1 的结果送到 Y 寄存器，第二阶段以 Y 的内容作为地址访问主存，并继续对 Y 寄存器加 1，结果送 PC，则也能实现 CALL 指令的功能。这种方式下，也是 9 个时钟周期。

4．假定某计算机字长16位，CPU内部结构如图5.5所示，CPU和存储器之间采用同步方式通信，按字编址。指令采用定长指令字格式，由两个字组成，第一个字指明操作码和寻址方式，第二个字包含立即数imm16。若一次存储器访问所用时间为两个时钟周期，每次存储器访问只能存取一个字，取指令阶段第二次访存将imm16取到MDR中。请写出下列指令在指令执行阶段（不考虑取指令阶段）的控制信号序列，并说明需要几个时钟周期。

（1）将立即数imm16加到寄存器R1中，此时imm16为立即操作数，即R[R1]←R[R1]+ imm16。

（2）将地址为imm16的存储单元的内容加到寄存器R1中，此时imm16为直接地址，即R[R1]←R[R1]+ M[imm16]。

（3）将存储单元imm16的内容作为地址，其所指的存储单元的内容加到寄存器R1中，此时imm16为间接地址，即：R[R1]←R[R1]+ M[M[imm16]]。

【分析解答】

图 5.5 所示的数据通路中，所有与内部总线相连的寄存器都有相应的 R_{in} 和/或 R_{out} 控制信号，以控制总线和寄存器之间的数据传送。总线和 ALU 输入端之间、Y 寄存器与 ALU 输入端之间都无需控制信号。ALU 输出与 Z 寄存器之间可以有控制信号 Z_{in}，也可以没有，此时，每来一个时钟，ALU 的输出总是被写入 Z 寄存器。为了明显表示 ALU 输出送 Z 寄存器，这里假定有控制信号 Z_{in}。

另外要说明的是，以下给出的是指令执行阶段的控制信号，因此，在执行阶段的开始，取指令阶段已经结束，此时，指令的第二个字（Imm16）已经从存储器中取出并被存放在 MDR 中。

（1）指令功能为 R[R1]←R[R1]+ Imm16 时，执行阶段不需要访存操作。因此，可用以下 3 个时钟周期完成。

MDR_{out}, Y_{in}

$R1_{out}$, add, (Z_{in})

Z_{out}, $R1_{in}$

（2）指令功能为 R[R1]←R[R1]+M[Imm16]时，执行阶段需要一次访存操作，因此，至少需要以下 5 个时钟周期。其中 $R1_{out}$, Y_{in} 这两个控制信号可以与 Read1 控制信息同时送出，

并在 Read2 操作阶段保持不变，也可以延迟到与 Read2 同时送出。

MDR_{out}, MAR_{in}

Read1,（$R1_{out}$, Y_{in}）

Read2, $R1_{out}$, Y_{in}

MDR_{out}, add, (Z_{in})

Z_{out}, $R1_{in}$

（3）指令功能为 R[R1]←R[R1]+M[M[Imm16]]时，执行阶段需要两次访存操作，因此，至少需要以下 8 个时钟周期。对 $R1_{out}$, Y_{in} 这两个控制信号的处理同（2）中一样。

MDR_{out}, MAR_{in}

Read1

Read2

MDR_{out}, MAR_{in}

Read1,（$R1_{out}$, Y_{in}）

Read2, $R1_{out}$, Y_{in}

MDR_{out}, add, (Z_{in})

Z_{out}, $R1_{in}$

5．某计算机字长16位，标志寄存器中的ZF、SF和OF分别是零标志、符号标志和溢出标志，采用双字节定长指令字。假定该计算机中有一条Bgt (大于零转移) 指令，其指令格式为：第一个字节指明操作码和寻址方式，第二个字节为偏移地址imm8，其功能是：

若 (ZF+(SF⊕OF)==0) 则 PC=PC+2+imm8 否则 PC=PC+2

完成如下要求并回答问题：

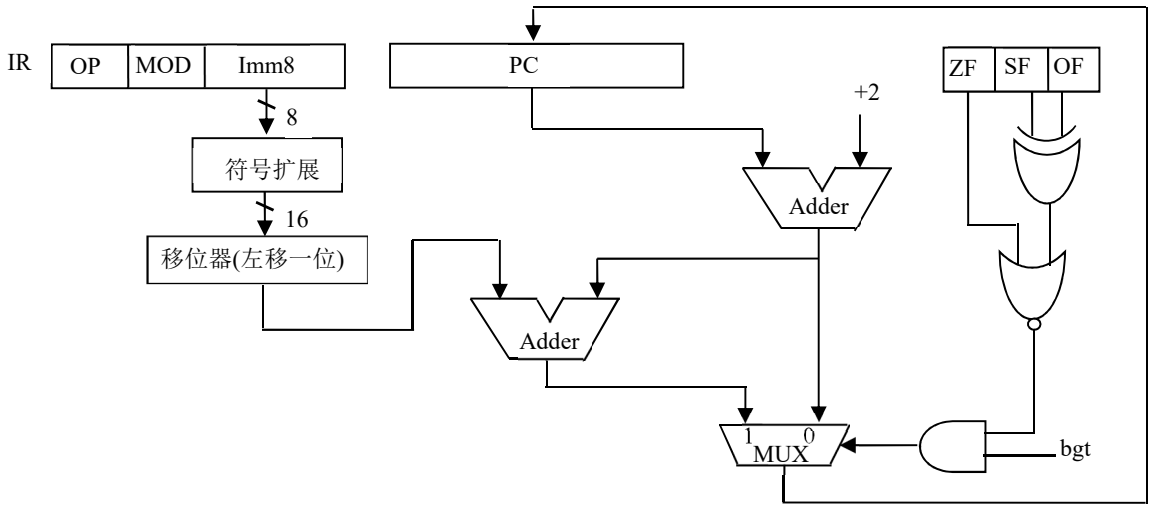
（1） 该计算机存储器的编址单位是多少位？

（2）画出实现Bgt指令的数据通路。

【分析解答】

（1）因为 PC 的增量是 2，且每条指令占 2 个字节，所以编址单位是字节。

（2）实现 bgt 指令的数据通路如下图所示。



若增加问题：bgt 指令执行的是带符号整数比较还是无符号整数比较？偏移地址 Imm8 的含义是什么？转移目标地址的范围是什么？

【分析解答】

根据“大于”的条件判断表达式(ZF+(SF⊕OF)==0)，可以看出该 bgt 指令实现的是带符号整数比较。因为无符号整数比较大小时，其判断表达式中应该有借位标志 CF，而没有溢出标志 OF 和符号标志 SF。

偏移地址 Imm8 为补码表示，说明转移目标指令可能在 bgt 指令之前，也可能在 bgt 指令之后。计算转移目标地址时，偏移量为 Imm8，而不是 Imm8×2，说明 Imm8 是相对地址，而不是相对指令条数。Imm8 占 8 位，即范围为−128～127，但因为采用双字节定长指令字，所以偏移量不可能是奇数，因此转移目标地址的范围是 PC+2+(−128)～PC+2+126，也即转移目标地址的范围是相对于 bgt 指令的前（负跳）126 个单元到后（正跳）128 个单元之间，用指令条数来衡量的话，就是相对于 bgt 指令的前 63 条指令到后 64 条指令之间。

6．假定图5.20给出的单周期数据通路对应的控制逻辑发生错误，使得控制信号RegWr、ALUASrc、Branch、Jump、MemWr、MemtoReg中某一个在任何情况下总是为0，则该控制信号为0时哪些指令不能正确执行（要求对题目中列出的每一个控制信号分别讨论）？

【分析解答】

若 RegWr=0，则所有需写结果到寄存器的指令（如：R-型指令、I-型运算类指令、load 指令等）都不能正确执行，因为寄存器不发生写操作。

若 ALUASrc=0，则 J 型指令（jal）可能不能正确执行，当指令中的目标寄存器 rd 不是 x0（0 号寄存器）时，需要将 PC+4（返回地址）存入 rd 中，但因为 ALUASrc=0，使得 PC 不能作为 ALU 的 A 输入端，因而在 ALU 中不能完成 PC+2 的计算。

若 Branch=0，则 branch 类（B 型）指令可能出错，因为永远不会发生跳转。

若 Jump=0，则 J 型指令（jal）可能出错，因为永远不会发生跳转。

若 MemWr=0，则 store 指令（sb、sh、sw）不能正确执行，因为存储器不能写入所需数据。

若 MemtoReg=0，则 load 指令（lb、lh、lw）发生错误，因为不能选择将存储器读出的内容送目的寄存器。

7．假定图5.20给出的单周期数据通路对应的控制逻辑发生错误，使得控制信号RegWr、ALUASrc、Branch、Jump、MemWr、MemtoReg中某一个在任何情况下总是为1，则该控制信号为1时哪些指令不能正确执行（要求对题目中列出的每一个控制信号分别讨

论）？

【分析解答】

若 RegWr=1，则所有无需写结果到寄存器的指令（如：store 指令、branch 类（B 型）指令等）都不能正确执行，因为寄存器发生了不该写结果的操作。

若 ALUASrc=1，则除了 J 型指令（jal）以外的需要在 ALU 中进行计算的指令都可能执行错误。

若 Branch=1，则除 branch 类（B 型）指令以外的指令可能出错，因为可能会发生不必要的跳转。

若 Jump=1，则除 J 型指令（jal）以外的指令都可能出错，因为会发生不必要的跳转。

若 MemWr=1，则除 store 指令（sb、sh、sw）以外的指令正确错误，因为存储器写入了不该写的数据。

若 MemtoReg=1，则除 load 指令（lb、lh、lw）以外的需要写 ALU 运算结果到寄存器的指令都会发生错误，因为选择了将存储器读出的内容送目的寄存器，而不是将 ALU 的结果送目的寄存器。

8．若要在RV32I指令集中增加一条swap指令（功能是交换两个寄存器的内容），可以有两种做法：一种做法是采用伪指令方式（即软件方式），这种情况下，当执行到swap指令时，用若干条已有指令构成的指令序列来代替实现；另一种做法是直接改动硬件来实现swap指令，这种情况下，当执行到swap指令时，则可在CPU上直接执行。

（1）写出用伪指令方式实现“swap rs, rt”时的指令序列（提示：伪指令对应的指令序列中不能使用其他额外寄存器，以免破坏这些寄存器的值）。

（2）假定用硬件实现swap指令时会使每条指令的执行时间增加10%，则swap指令要在程序中占多大的比例才值得用硬件方式来实现？

（3）采用硬件方式实现时，在不对通用寄存器组进行修改的情况下，能否在单周期数据通路中实现swap指令？对于多周期数据通路的情况又怎样？

【分析解答】

（1）若在伪指令“swap rs, rt”的指令序列中使用除 rs 和 rt 以外的额外寄存器，则额外寄存器的内容会被指令序列破坏，因此，伪指令的指令序列中一般不能用额外寄存器。伪指令“swap rs, rt”的指令序列包含以下三条指令，没有用到额外寄存器。

```
xor  rs, rs, rt
xor  rt, rs, rt
xor  rs, rs, rt
```

（2）假定“swap rs, rt”指令所占比例为 x（$0 \le x \le 1$），其他指令比例为 $1-x$，则用硬件实现该指令时，程序执行时间为原来的 $1.1\times(x+1-x)=1.1$ 倍。用软件实现该指令时，程序执行时间为原来的 $3x+1-x=2x+1$ 倍。因此，当 $1.1 < 2x+1$ 时，硬件实现才有意义，由此

可知，x > 0.05，也即：当“swap rs, rt”指令在程序中的比例大于 5%时，才值得用硬件方式来实现该指令。

（3）在单周期数据通路中，所有指令的执行都在一个时钟周期内完成，数据总是在时钟边沿被写入寄存器堆，即本条指令执行的结果总是下条指令开始（即下个时钟到来）时，才开始被写到寄存器堆中，因此一个时钟周期只能写一次寄存器。而 swap 指令执行过程中需要多次写寄存器，在不对寄存器堆进行修改的情况下，无法在单周期数据通路中实现 swap 指令；对于多周期数据通路，一个指令周期可以有多个时钟周期，因而可以多次写寄存器，因此，在不对寄存器堆进行修改的情况下，swap 指令可以在多周期数据通路中实现。

10．假定图5.33所示的多周期数据通路对应的有限状态机控制器发生错误，使得控制信号PCWr、MARWr、RegWr、BMUX、PCout中某一个在任何情况下总是为0，则该控制信号为0时哪些指令不能正确执行（要求对题目中列出的每一个控制信号分别讨论）？

【分析解答】

若 PCWr=0，则所有指令都不能正确执行，因为无法正确地更新 PC。

若 MARWr=0，则 Load 和 Store 指令不能正确执行，因为加法器计算得到的主存地址无法写入 MAR，因而也就不能将正确的地址通过系统总线传送到主存进行指定单元的数据读写。

若 RegWr=0，则所有需要写结果到寄存器的指令（如 R-型指令、I-型运算指令、Load 指令）都不能正确执行，因为寄存器不发生写操作。

若 BMUX=0，则 R-型指令执行错误，因为第 2 个寄存器中的操作数无法选择作为 ALU 输入端 B 的输入。

若 PCout=0，则所有指令都不能正确执行，因为取指令时无法将正确的指令地址通过系统总线传送到主存进行指令的读取。

11．假定图5.33所示的多周期数据通路对应的有限状态机控制器发生错误，使得控制信号PCWr、MARWr、RegWr、BMUX、PCout中某一个在任何情况下总是为1，则该控制信号为1时哪些指令不能正确执行（要求对题目中列出的每一个控制信号分别讨论）？

【分析解答】

若 PCWr=1，则每个时钟都会更新 PC，从而可能导致计算出的下条指令地址不正确。

若 MARWr=1，则因为每个周期都会修改 MAR，因此所有指令都可能出错。

若 RegWr=1，则所有不需要写结果到寄存器的指令（如 Store 指令）都不能正确执行，因为有寄存器写入了不该写的数据。

若 BMUX=1，则 I-型运算类指令和 Load/Store 指令执行错误，因为立即数 imm16 扩展后的结果无法选择作为 ALU 输入端 B 的输入。

若 PCout=1，则 Load 和 Store 指令不能正确执行，因为每个时钟周期都会将 PC 的内容发送到地址总线上，当执行 Load/Store 指令需要根据 MAR 的内容读写主存时，PC 和 MAR 两个寄存器的内容同时会传送到地址总线上，从而发生冲突。

14．假定在一个5级流水线（如图5.43所示）处理器中，各主要功能单元的操作时间为：存储单元--200ps；ALU和加法器--150ps；通用寄存器组的读口或写口--50ps。请问：

（1） 若执行阶段EX所用的ALU操作时间缩短20%，则能否加快流水线执行速度？如果能的话，能加快多少？如果不能的话，为什么？

（2） 若ALU操作时间增加20%，对流水线的性能有何影响？

（3） 若ALU操作时间增加40%，对流水线的性能又有何影响？

【分析解答】

（1）ALU 操作时间缩短 20%不能加快流水线指令速度。因为指令流水线的执行速度取决于最慢的功能部件所用时间，最慢的是存储器，只有缩短了存储器的操作时间才可能加快流水线速度。

（2）ALU 操作时间延长 20%时，变为了 180ps，比存储器所用时间 200ps 还小，因此，对流水线性能没有影响；

（3）ALU 操作时间延长 40%时，变为了 210ps，比存储器所用时间 200ps 大，因此，在不考虑流水段寄存器延时的情况下，流水线的时钟周期从 200ps 变为 210ps，流水线执行速度降低了（210-200)/200=5%。

16．假定最复杂的一条指令所用的组合逻辑分成六部分，依次为A~F，其延迟分别为80ps、30ps、60ps、50ps、70ps、10ps。在这些组合逻辑块之间插入必要的流水段寄存器就可实现相应的指令流水线，寄存器延迟为20ps。理想情况下，以下各种方式所得到的时钟周期、指令吞吐率和指令执行时间各是多少？应该在哪里插入流水段寄存器？

（1）插入一个流水段寄存器，得到一个两级流水线。

（2）插入两个流水段寄存器，得到一个三级流水线。

（3）插入三个流水段寄存器，得到一个 4 级流水线。

（4）吞吐量最大的流水线。

【分析解答】

（1）两级流水线的平衡点在 C 和 D 之间，其前面一个流水段的组合逻辑延时为 80+30+60=170ps，后面一个流水段的组合逻辑延时为 50+60+20=130ps。最长功能段延时为 170ps，加上流水段寄存器延时 20ps，因而时钟周期为 190ps，理想情况下，指令吞吐率为每秒钟执行 1/190ps=5.26G 条指令。每条指令在流水线中的执行时间为 2×190=380ps。

（2）两个流水段寄存器分别插在 B 和 C、D 和 E 之间，这样第一个流水段的组合逻辑延时为 80+30=110ps，中间第二段的延时为 60+50=110ps，最后一个段延时为 60+20=80ps。

这样，每个流水段所用时间都按最长延时调整为 110+20=130ps，故时钟周期为 130ps，指令吞吐率为每秒钟执行 1/130ps=7.69G 条指令，每条指令在流水线中的执行时间为 3×130=390ps。

（3）三个流水段寄存器分别插在 A 和 B、C 和 D、D 和 E 之间，这样第一个流水段的组合逻辑延时为 80ps，第二段延时为 30+60=90ps，第三段延时为 50ps，最后一段延时为 60+20=80ps。这样，每个流水段都以最长延时调整为 90+20=110ps，故时钟周期为 110ps，指令吞吐率为每秒钟执行 1/110ps=9.09G 条指令，每条指令在流水线中的执行时间为 4×110=440ps。

（4）因为各功能部件对应的组合逻辑中最长延时为 80ps，所以，流水线的时钟周期肯定比 80ps+20ps=100ps 长。为了达到最大吞吐率，时钟周期应该尽量短，因此，最合理的划分方案应该按照每个时钟周期为 100ps 来进行。根据每个功能部件所用时间可知，流水线至少按 5 段来划分，分别把流水线寄存器插入在 A 和 B、B 和 C、C 和 D、D 和 E 之间，这样各段的组合逻辑延时为 80ps、30ps、60ps，50ps 和 80ps。其中，最后一个延时 80ps 是 E 和 F 两个阶段的时间相加而得到的。这样时钟周期为 100ps，指令吞吐率为每秒钟执行 1/100ps=10G 条指令，每个指令的执行时间为 5×100=500ps。

通过对上述 4 种情况进行分析，可以得出以下结论：划分的流水段多，时钟周期就变短，指令执行吞吐率就变高，而相应的额外开销（即插入的流水段寄存器的延时）也变大，使得一条指令的执行时间变长。

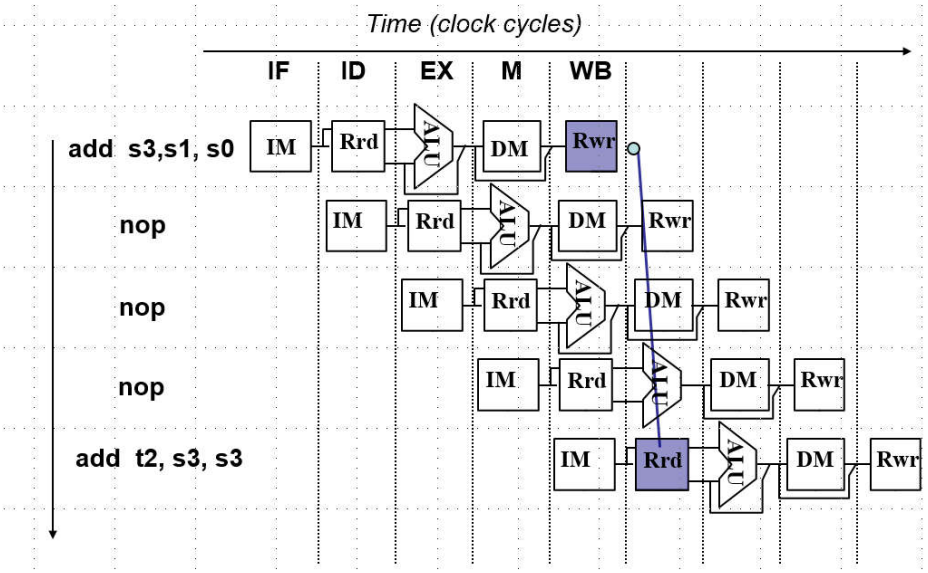
17．以下指令序列中，哪些指令对之间发生数据相关？假定采用“取指、译码/取数、执行、访存、写回”5段流水线方式，那么不用“转发”技术的话，需要在发生数据相关的指令前加入几条nop指令才能使这段程序避免数据冒险？如果采用“转发”是否可以完全解决数据冒险？不行的话，需要在发生数据相关的指令前加入几条nop指令才能使这段RV32I程序不发生数据冒险？

```
add   s3, s1, s0
add   t2, s3, s3
lw    t1, 0(t2)
add   t3, t1, t2
```

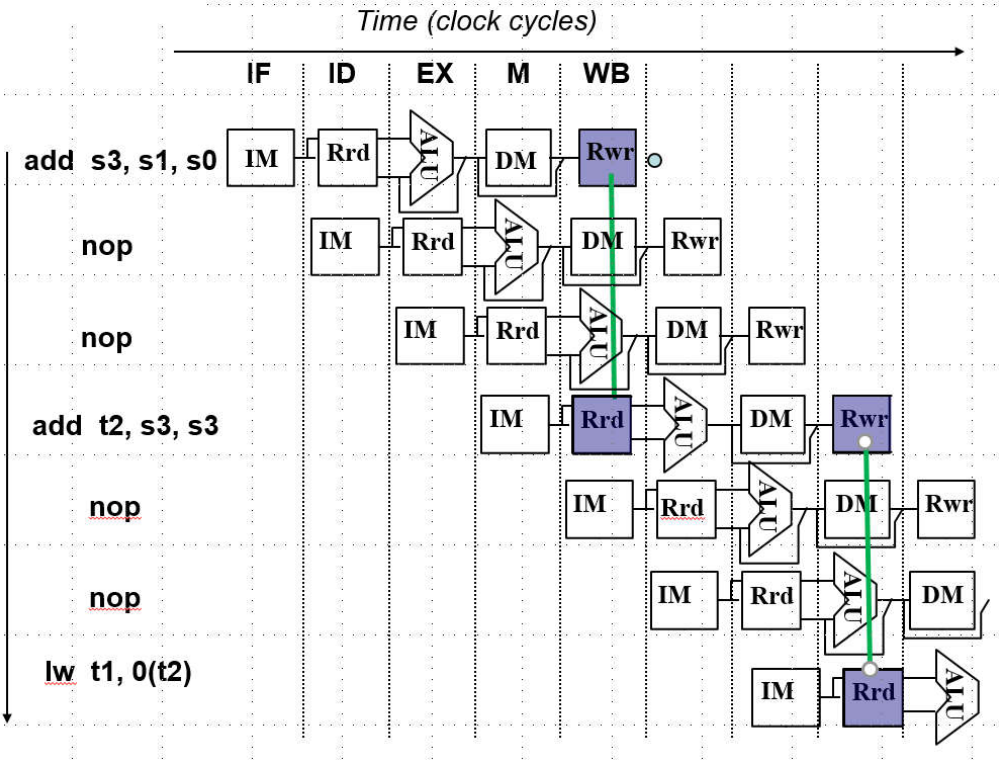
【分析解答】

第 1 和第 2 条指令、第 2 和第 3 条指令、第 2 条和第 4 条指令、第 3 条和第 4 条指令之间发生数据相关。

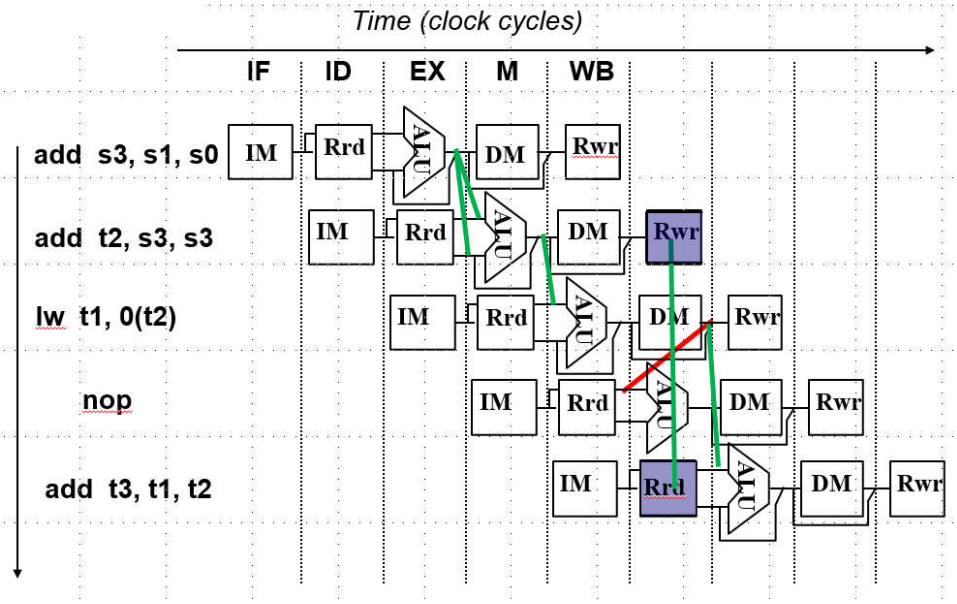
如下图所示，在一个 5 段流水线处理器中，若不采用“转发”技术，则为避免相邻两条指令之间的数据冒险，需要在两条指令之间插入 3 条 nop 指令。



不过，通常在 WB 阶段的前半周期能够写入寄存器，在相邻的后 1 条指令的 ID 阶段的后半周期可以读到寄存器中的新数据（上半周期写，下半周期读），因而在相邻两条数据相关的指令之间插入两条 nop 指令即可。因此，为避免数据冒险，在上述指令序列中，需在前 3 条指令后各插入 2 条 nop 指令（如下图所示）。



可以采用“转发”技术解决第 1 和第 2 条指令、第 2 和第 3 条指令、第 2 和第 4 条指令之间的数据冒险；但是，因为第 3 和第 4 条指令之间是 Load-use 数据冒险，因而不能通过转发技术解决（参见下图中的红线），需要在第 3 条指令后加一条 nop 指令（如下图所示）。



18．假定以下RV32I指令序列在图5.43所示的流水线数据通路中执行：

```
add   s3, s1, s0
sub   t2, s3, s3
lw    t1, 0(t2)
add   t3, t1, t2
add   t1, s4, s5
```

问答以下问题。

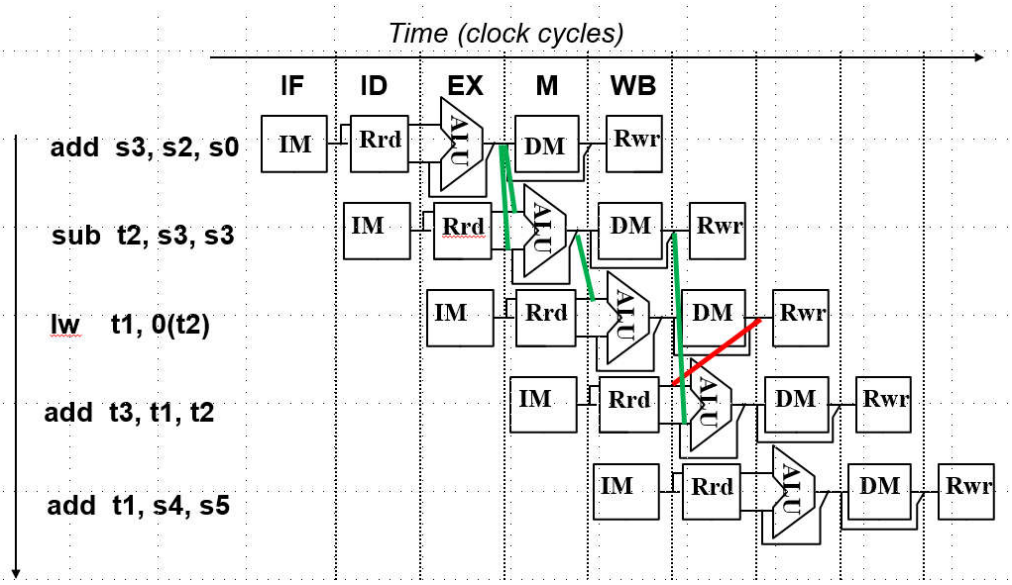
（1）上述指令序列中，哪些指令的哪个寄存器需要转发，转发到何处？

（2）上述指令序列中，是否存在 Load-use 数据冒险？

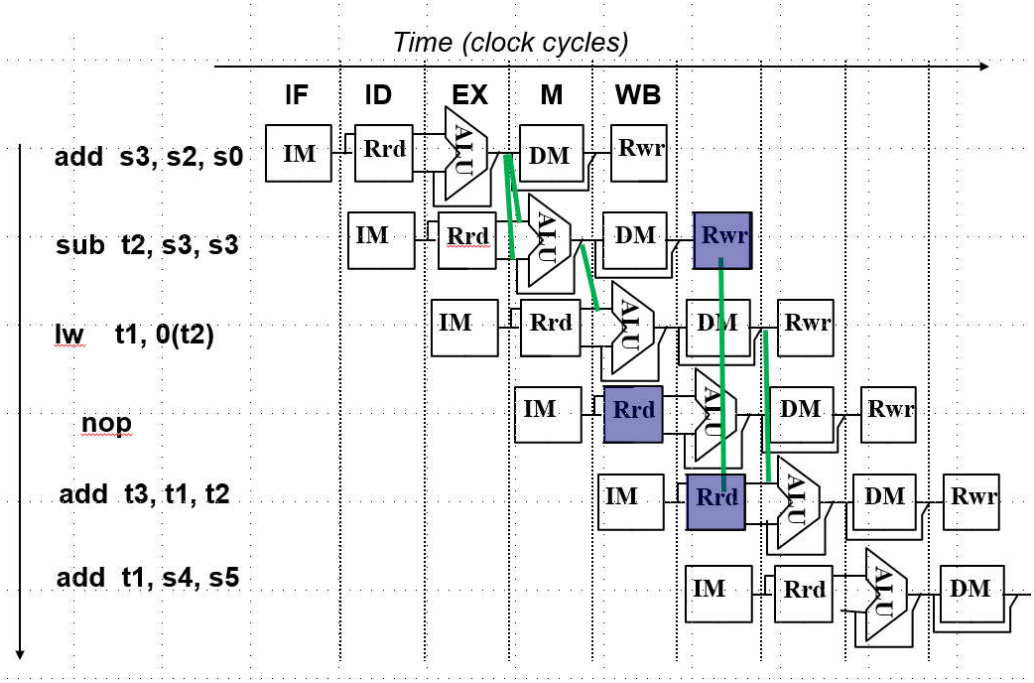
（3）第 5 周期结束时，各指令执行状态是什么？哪些寄存器的数据正被读出？哪些寄存器将被写入？

【分析解答】

（1）如下图所示，第 1 条指令的目的寄存器 s3 需要转发，直接从 EX/M 流水段寄存器转发到下条指令的 ALU 两个输入端。第 2 条指令的目的寄存器 t2 需要从 EX/M 流水段寄存器转发到第 3 条指令的 ALU 输入端。第 3 条指令的目的寄存器 t1 需要从 M/WB 流水段寄存器转发到第 4 条指令的 ALU 输入端，但由于来不及转发（如红线所示），因此需要在第 3 条指令后加入一条 nop 指令。



（2）第 3 条和第 4 条指令之间是 Load-use 数据冒险。在第 3 条指令后需加一条 nop 指令。加入 nop 指令后的执行情况如下图所示。



（3）如上图所示，在第 5 个时钟周期结束时，各条指令的执行情况如下：

第 1 条指令执行完“写回（WB）”阶段，寄存器 s3 将被写入；

第 2 条指令执行完“访存（M）”阶段，sub 指令在该阶段进行的是空操作，上一个时钟周期中的执行结果（在流水段寄存器 EX/M 中的 ALU 输出结果）将被写入 M/WB 流水段寄存器；

第 3 条指令执行完“执行（EX）”阶段，ALU 输出段输出的主存地址将被写入 EX/M 流水段寄存器中；

由于在第 3 条指令后加入了一条 nop 指令，因此，第 4 条指令延迟了一个周期执行，因此刚执行完“取指（IF）”阶段，取出的指令将被写入 IF/ID 流水段寄存器中。

第 5 条指令还未进入流水线。

19．假定有一个程序的指令序列为“lw, add, lw, add, …”。add指令仅依赖它前面的lw指令，而lw指令也仅依赖它前面的add指令，寄存器写口和寄存器读口分别在一个时钟周期的前、后半个周期内独立工作。请问：

（1）在带转发的 5 段流水线中执行该程序，其 CPI 为多少？

（2）在不带转发的 5 段流水线中执行该程序，其 CPI 为多少？

【分析解答】

（1）若流水线中有“转发”逻辑，并且寄存器写口和寄存器读口分别在一个时钟周期的前、后半个周期内工作，则只存在 lw 指令和 add 指令之间的一个 load-use 数据冒险，因此，每个 lw 指令和 add 指令之间有一次流水线阻塞，即每一对 lw 和 add 指令需要用 3 个时钟周期完成，因而 CPI 为 1.5。

（2）若流水线中没有“转发”逻辑，而寄存器写口和寄存器读口分别在一个时钟周期的前、后半个周期内工作，则在每两条相邻指令之间都会有两个周期的阻塞，这样每条指令相当于都要有 3 个时钟周期才能完成，因而 CPI 为 3。

20．假定在一个带转发的5段流水线中执行以下RV32I程序段，应怎样调整指令序列使其性能达到最好？

```
lw      s2, 100(s6)
add     s2, s2, s3
lw      s3, 200(s7)
add     s6, s4, s7
sub     s3, s4, s6
lw      s2, 300(s8)
beq     s2, s8, Loop
```

【分析解答】

因为采用“转发”技术，所以，只要对 load-use 数据冒险进行指令序列调整。从上述指令序列来看，第 1 和第 2 条指令、第 6 和第 7 条指令之间存在 load-use 数据冒险，所以，可将与第 2 和第 3 条指令无关的第 4 条指令插入第 2 条指令之前；将与第 6 和第 7 条无关的第 5 条指令插入第 7 条指令之前。调整顺序后的指令序列如下（红字部分为变换了位置的指令）。

```
1       lw      s2, 100(s6)
4       add     s6, s4, s7
2       add     s2, s2, s3
3       lw      s3, 200(s7)
```

```
6        lw    s2, 300(s8)
5        sub   s3, s4, s6
7        beq   s2, s8, Loop
```

21．在一个采用“取指、译码/取数、执行、访存、写回”的5段流水线中，若检测相减结果是否为“零”的操作在执行阶段进行，则分支延迟损失时间片（即分支延迟槽）为多少？以下一段RV32I指令序列中，在考虑数据转发的情况下，哪些指令执行时会发生流水线阻塞？各需要阻塞几个时钟周期？

```
loop:  add  t1, s3, s3
       add      t1, t1, t1
       add      t1, t1, s6
       lw       t0, 0(t1)
       bne      t0, s5, Exit
       add      s3, s3, s4
       j    Loop
Exit:
```

【分析解答】

在书中图 5.43 所示的 5 段流水线中，检测结果是否为“零”并更新 PC 的操作在“访存（M）”阶段进行，这种情况下，分支延迟损失时间片（分支延迟槽）为 3。

若检测结果是否为“零”并更新 PC 的操作提前到在“执行（EX）”阶段进行，则分支延迟损失时间片（分支延迟槽）变为 2。

在采用数据转发技术的情况下，上述指令序列中，第 4 条和第 5 条指令之间为 Load-use 冒险，无法通过转发技术解决，此时第 5 条 bne 指令的执行被阻塞一个时钟。第 5 条指令是分支指令 bne，由于该指令执行时条件检测结果不能马上得到，因此其后指令的执行被阻塞，阻塞的时钟周期数等于分支延迟损失时间片。第 7 条指令为无条件跳转指令 jal（j loop 是其伪指令），假定“j loop”指令更新 PC 的操作在“执行（EX）”阶段进行，则其后指令的执行将被阻塞 2 个时钟周期；假定更新 PC 的操作在“译码（ID）”阶段进行，则被阻塞 1 个时钟周期。

22．假设数据通路中各主要功能单元的操作时间如下：存储单元，200ps；ALU和加法器，100ps；通用寄存器组的读口或写口，50ps。程序中指令的组成比例如下：取数，25%；存数，10%；ALU，52%；分支，11%；跳转，2%。假设时钟周期取存储操作时间的一半，MUX、控制单元、PC、扩展器和传输线路等的延迟都忽略不计，则下面的实现方式中，哪个更快？快多少？

（1）单周期方式：每条指令在一个固定长度的时钟周期内完成；

（2）多周期方式：每类指令的 CPI 为取数-7，存数-6，ALU-5，分支-4，跳转-4；

（3）流水线方式：采用取指 1、取指 2、取数/译码、执行、存取 1、存取 2、写回 7 段流水线；没有结构冒险；数据冒险采用“转发”技术处理；load 指令与后续各指令之间存在依赖关系的概率分别 1/2、1/4、1/8、…；分支延迟损失时间片为 2，预测准

确率为 75%；不考虑异常、中断和访问缺失引起的流水线冒险。

【分析解答】

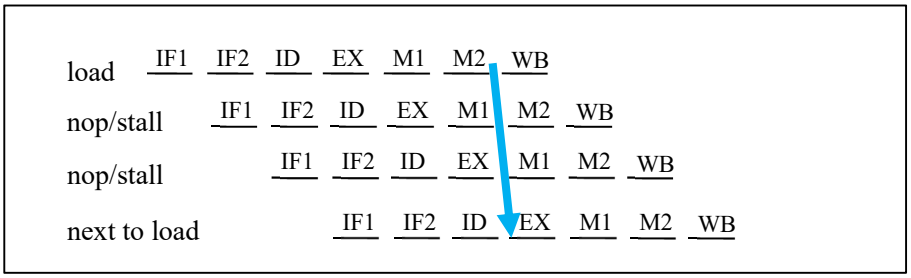
（1）单周期方式下，时钟周期为 200+50+100+200+50=600ps，故一条指令的执行时间为 600ps。

（2）多周期方式下，CPI=0.25×7+0.10×6+0.52×5+0.11×4+0.02×4 = 5.47，存储器操作变为在两个时钟周期内完成后，多周期数据通路的时钟周期为 100ps，故平均一条指令的执行时间为 100×5.47=547ps。

（3）流水线方式下，存储器操作变为在两个时钟周期内完成后，其流水线包含了 7 个阶段。

对于分支指令，若预测正确，则不需额外时钟周期，故只需 1 个时钟周期；若预测错误，则因为分支延迟损失时间片为 2，所以应该将错误预取的 2 条指令冲刷掉，额外多用了 2 个时钟周期，因此，预测错误时共需 3 个时钟周期，故分支指令的 CPI=0.25×3+0.75×1=1.5。

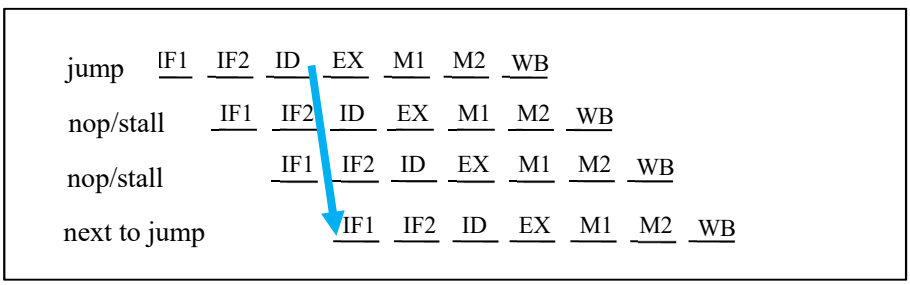
对于 load 指令，因为一个存储操作占用两个时钟周期（M1、M2），所以随后第 1 条指令则需 3 个（其中阻塞 2 个）时钟周期（如下图所示）；随后第二条指令需 2 个（其中阻塞 1 个）时钟周期，以后的指令都不需要阻塞，故 CPI=1/2×3+1/4×2+2/8×1=2.25。



对于 ALU 指令，随后的数据相关指令都可通过转发解决，故 CPI=1。

对于 store 指令，不会发生数据冒险，故 CPI=1。

对于 jump 指令，最快也要在译码（ID）阶段才能确定转移地址，因此需阻塞 2 个时钟周期，加上本身一个时钟周期，共 3 个时钟周期（如下图所示），故 CPI=3。



综合 CPI=0.25×2.25+0.10×1+0.52×1+0.11×1.5+0.02×3=1.41。因此，平均一条指令的执行时间为 1.41×100=141ps。

由上述分析可知，流水线处理器的指令执行速度最快，是单周期的 600/141=4.26 倍，

是多周期的 547/141=3.844 倍。

23．有一段程序的核心模块中有5条分支指令，该模块将会被执行成千上万次，在其中一次执行过程中，5条分支指令的实际执行情况如下（T：Taken；N：not Taken）。

分支指令 1：T – T – T。

分支指令 2：N – N – N – N。

分支指令 3：T – N –T – N – T– N。

分支指令 4：T – T – T– N – T。

分支指令 5：T – T – N – T– T – N – T。

假定各个分支指令在每次模块执行过程中实际执行情况都一样，并且动态预测时，每个分支指令都有自己的预测表项，每次执行该模块时的初始预测位都相同。请分析并给出以下几种预测方案的预测准确率。

（1） 静态预测，总是预测转移 （Taken）。

（2） 静态预测，总是预测不转移（not Taken）。

（3） 一位动态预测，初始预测转移 （Taken）。

（4） 二位动态预测，初始预测弱转移（Taken）。

【分析解答】

预测准确率=预测正确次数/总预测次数×100%。以下 R 表示正确预测次数，W 表示错误预测次数。

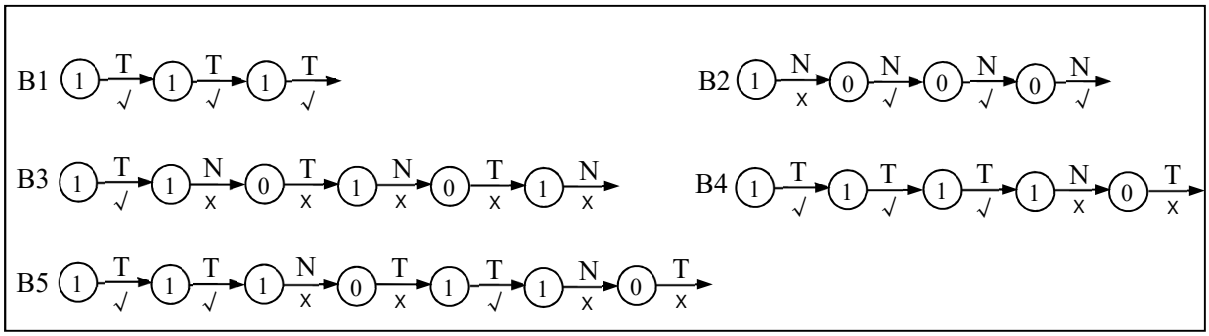
（1）B1: R-3, W-0； B2: R-0, W-4； B3: R-3, W-3； B4: R-4, W-1； B5: R-5, W-2；60%

（2）B1: R-0, W-3； B2: R-4, W-0； B3: R-3, W-3； B4: R-1, W-4； B5: R-2, W-5；40%

（3）根据图 5.59 所示的状态转换图，可以得到各分支点的预测情况如下：

B1: R-3, W-0； B2: R-3, W-1； B3: R-1, W-5； B4: R-3, W-2； B5: R-3, W-4；52%

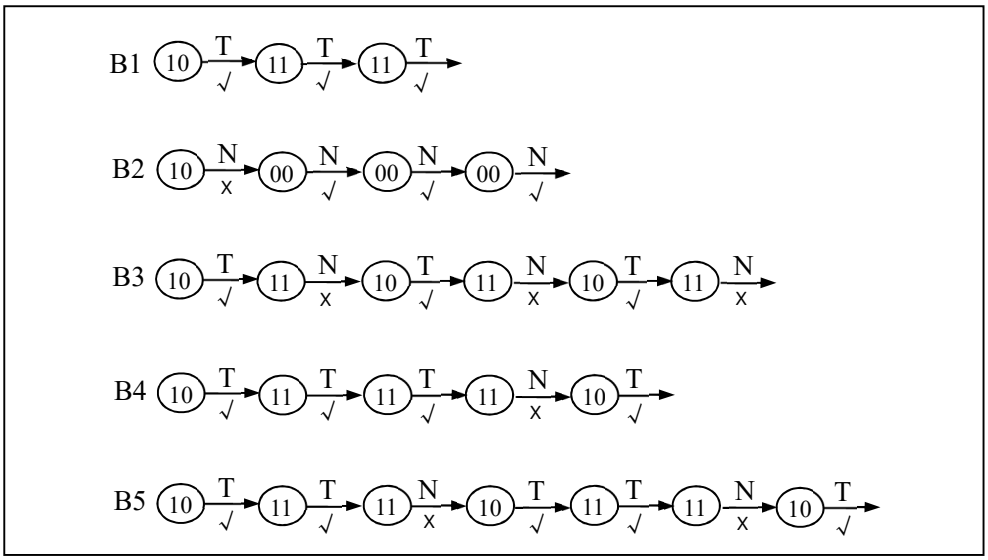
具体每次的预测结果如下图所示。



（4）根据图 5.61 所示状态转换图，可以得到各分支点的预测情况如下：

B1: R-3, W-0； B2: R-3, W-1； B3: R-3, W-3； B4: R-4, W-1； B5: R-5, W-2；72%

具体每次的预测结果如下图所示。


B1
10
T
√
11
T
√
11
T
√
B2
10
N
x
00
N
√
00
N
√
00
N
√
B3
10
T
√
11
N
x
10
T
√
11
N
x
10
T
√
11
N
x
B4
10
T
√
11
T
√
11
T
√
11
N
x
10
T
√
B5
10
T
√
11
T
√
11
N
x
10
T
√
11
T
√
11
N
x
10
T
√