计算机系统结构第一次实验报告

一、实验环境：

使用模拟器 A，即课程自带的 MIPSsim 模拟器完成实验

二、实验过程：

1. 实现无冲突的流水线场景：

代码编写：

```
.text
main:
ADDIU $r1, $r0, ID
ADD $r7, $r7, 1
ADD $r8, $r8, 1
LW $r3, 0($r1)
LB $r4, 0($r1)
LBU $r5, 0($r1)
ADD $r3, $r3, 1111
SW $r4, 0($r1)
TEQ $r0, $r0

.data
ID:
.word 3172
```

程序分析：

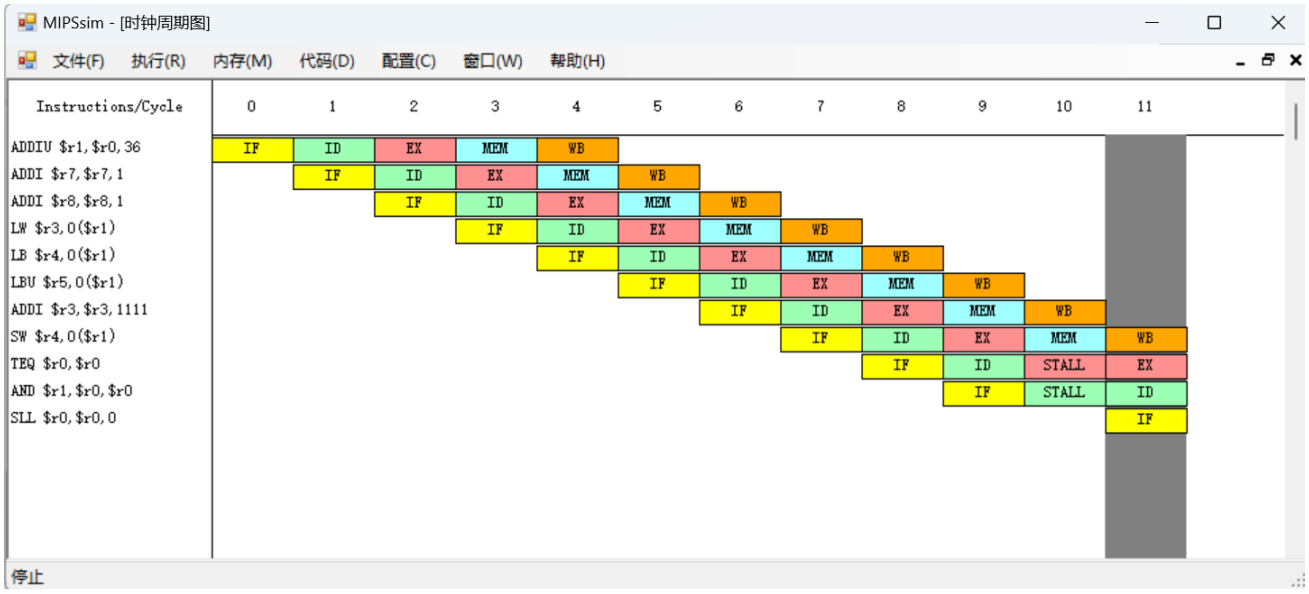
定义 ID 为一个 WORD 长度的整数，程序完成的功能是将 ID 按照 16 位有符号数、8 位有符号数、8 位无符号数加载到寄存器中，为了防止写后读 RAW 冲突，在计算 ID 的地址之后，插入了两条 ADD 指令。加载到相应的寄存器之后，对字型整数 ID 加上 1111，再将其低 8 位的值写回到内存。

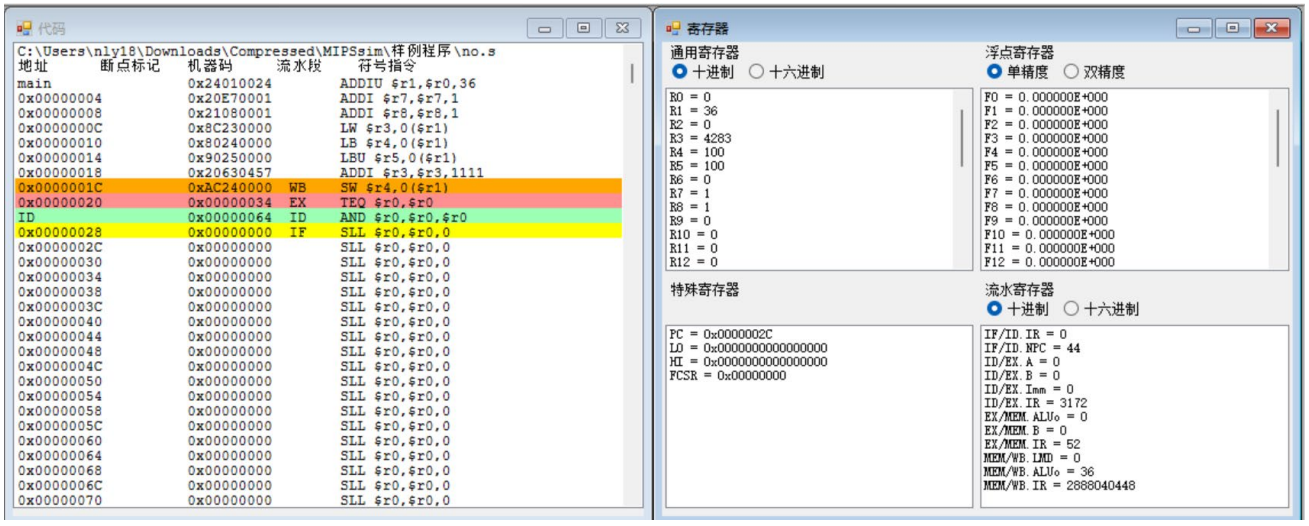
流水线分析：

程序中不包含写后读 RAW、读后写 WAR、写后写 WAW 冲突，也不包含

控制冲突，按照软件的假定，分别有一个加法、乘法和除法部件，不会造成结构冲突。

执行情况：





程序以 TEQ 指令为结尾，在之前的所有指令执行中没有出现流水线停顿，说明没有冲突发生。

性能分析：(假设周期长度为 t)

吞吐率 TP=8/(12t)=2/(3t)

加速比 S=TP*5t=10/3

效率 E=S/5=2/3=66.7%

2. 开启定向功能，加入某些写后读 RAW 相关：

代码编写：

```
.text
main:
ADDIU $r1, $r0, ID
LW $r3, 0($r1)
LB $r4, 0($r1)
ADD $r3, $r3, 1111
SW $r3, 0($r1)
TEQ $r0, $r0

.data
ID:
.word 3172
```

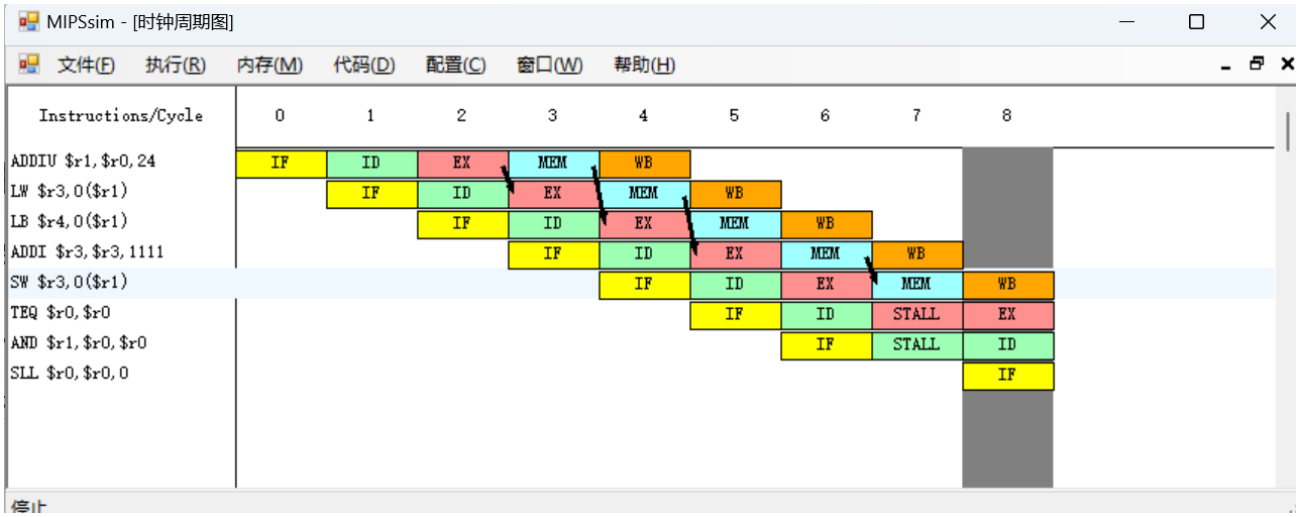
程序分析：

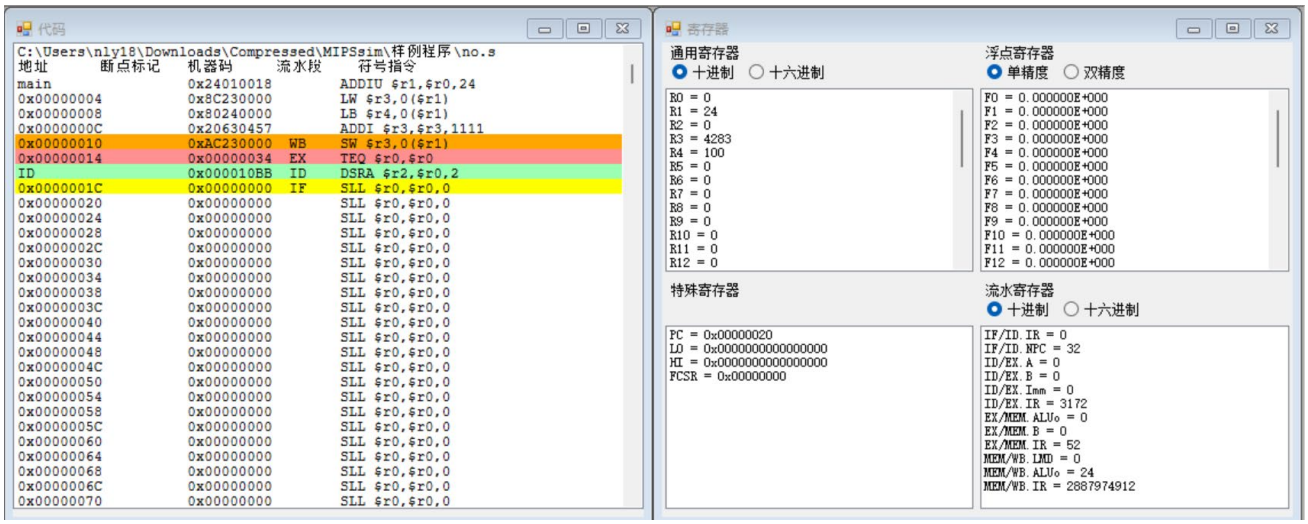
去掉了 ADDIU 指令之后的两个 ADD 指令，与接下来的两条 load 指令造成两次 RAW 相关；删去了 LBU 指令，造成 LW 与之后的 ADD 指令的一次 RAW 相关；ADD 之后将 R3 中的数字立即送入内存，造成一次 RAW 相关。由于 5 段流水线不存在 WAR 相关和 WAW 相关，故无法展示这两种相关。

流水线分析：

按照定向技术推断，ADDIU 之后的两个 RAW 相关，分别会由 EX/MEM 锁存器和 MEM/WB 锁存器传给两条 load 指令的 EX 段执行; LW 与 ADD 之间的 RAW 冲突也会从 LW 的 MEM/WB 锁存器传入到 ADD 的 EX 段执行, SW 指令只有在 MEM 段才需要 rs 寄存器的值，故 ADD 指令的 MEM/WB 锁存器会将 R3 的值传入 SW 的 MEM 段。

执行情况：





程序中的 RAW 冲突均由定向技术完全解决，且分析没有遗漏，执行过程中没有发生停顿。

性能分析：(假设周期长度为 t)

吞吐率 TP=5/(9t)

加速比 S=TP*5t=25/9

效率 E=S/5=5/9=55.6%

3. 关闭定向功能，加入写后读 RAW 相关：

代码编写：

与开启定向功能的程序相同，不再展示。

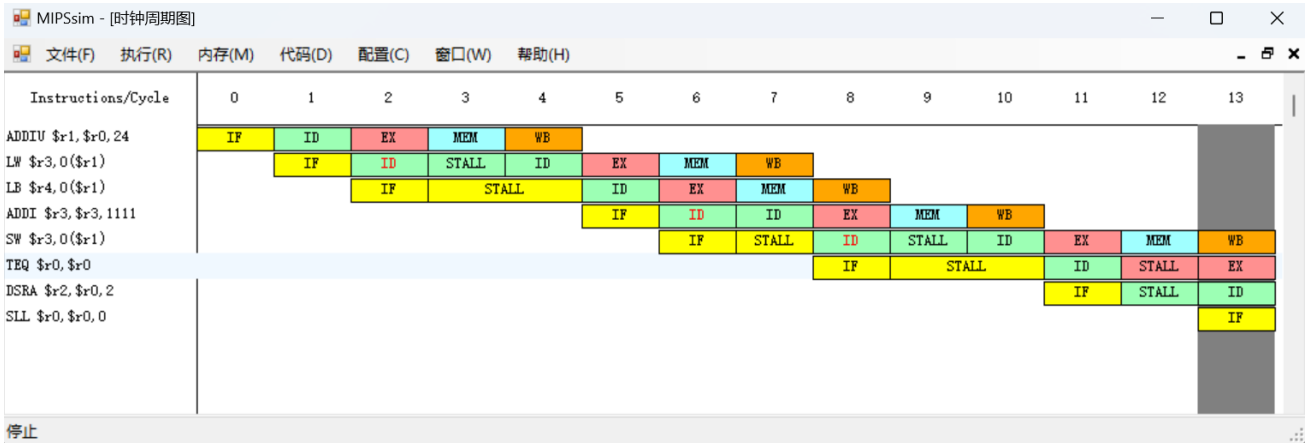
程序分析：

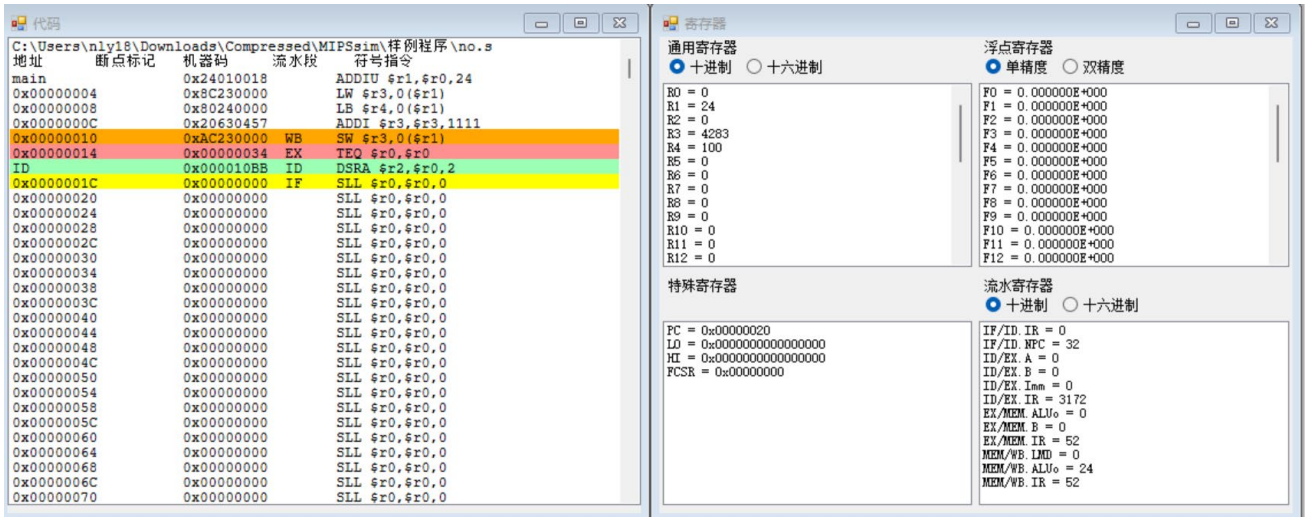
由于会发生 RAW 相关，可能会有相应的结构相关出现（停顿之后多个指令抢占部件的现象）

流水线分析：

没有开启定向技术时，在 LW 指令完成译码之后，发现 R3 寄存器与上一条指令发生了 RAW 冲突，故引起了流水线停顿，插入一个 STALL，之后重新开始译码，并在进入 EX 段时 ADDIU 指令已经将 R3 的内容写回寄存器文件；LB 指令与 ADDIU 指令也存在 RAW 冲突，由于 LW 的 STALL 将整个流水线停顿，LB 也收到一个停顿，停顿结束后 LW 进入 ID 段，LB 也想要进入 ID 段，于是等候，再次插入一个 STALL；LW 之后的 ADD 指令在 LB 指令的两个 STALL 之后的周期开始 IF，进入 ID 段之后发现 R3 寄存器发生 RAW 相关，故引起一个周期的停顿，在下一个周期再次译码，能够在 EX 段读取到 LW 在上个周期写回的数据；由于 ADD 指令 ID 段的停顿，SW 指令在 IF 段之后也要插入一个 STALL 停顿避免结构相关，在 ID 完成之后，发现需要的数据与上一条指令 ADD 有 RAW 相关，故再次插入一个 STALL，等 ADD 的 WB 完成之后进入 EX 段。

执行情况：





上述流水线分析中的 RAW 冲突均有发生，且执行情况符合分析。

性能分析：(假设周期长度为 t)

吞吐率 TP=5/(14t)

加速比 S=5*5/14=25/14

效率 E=S/5=5/14=35.7%

4. 控制相关：

代码编写：

```
.text
main:
ADDIU $r1, $r0, ID
LW $r3, 0($r1)
```

```
LB $r4, 0($r1)
BEQ $r5, $r5, PROG
NOP

PROG:
ADD $r3, $r3, 1111
SW $r3, 0($r1)
TEQ $r0, $r0

.data
ID:
.word 3172
```

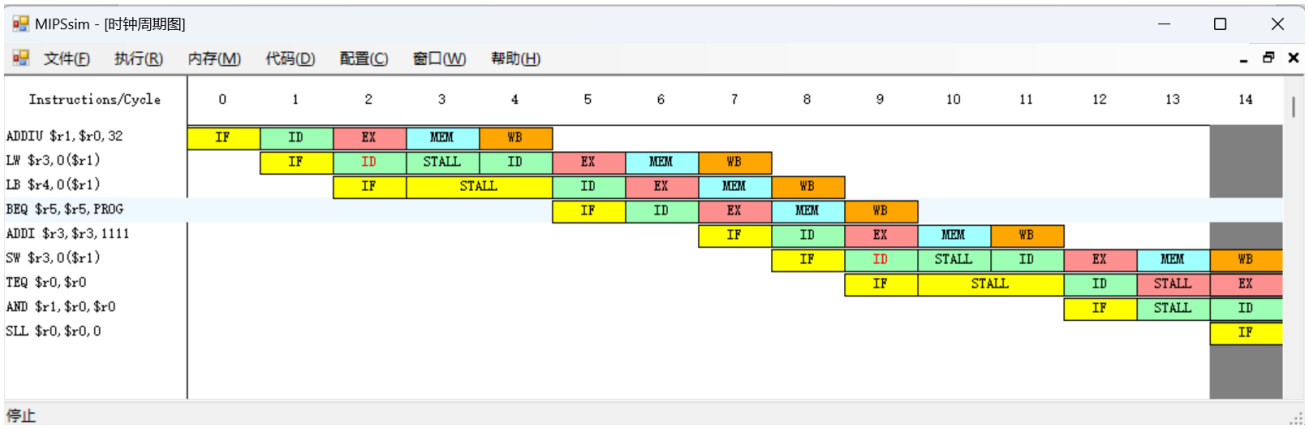
程序分析：

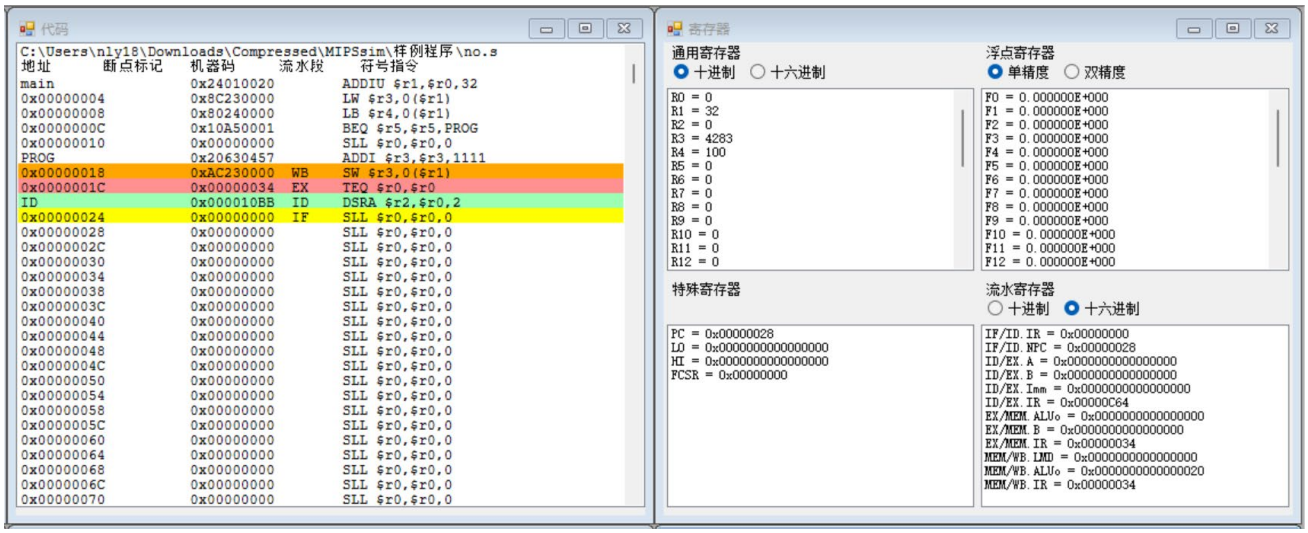
程序中加入一个 BEQ 条件转移指令，在本程序中检测 R5 与 R5 的值是否相等，故一定发生跳转，其余的指令均与上一种情况编写的程序相同。

流水线分析：

在 BEQ 的 ID 段中，会检测两个寄存器是否值相等，并且将正确的目标地址放入 ID.NPC 中，所以会引起下一条指令的停顿，延缓一个周期进行 IF。

执行情况：





可以看到 BEQ 的 ID 段完成之后才会发生目标指令 ADD 的 IF。

性能分析：(假设周期长度为 t)

吞吐率 TP=6/(15t)=2/(5t)

加速比 S=TP*5t=2

效率 E=S/5=2/5=40%

三、实验感悟：

5 段流水线中不存在 WAW、WAR 相关，但是存在 RAW 相关，且会按照不同的执行情况发生 STALL 的插入或者指令的延缓 Fetch；此外还存在 BEQ 这样的指令引起的控制相关和同时多个指令争抢部件引起的结构相关。由于流水线气泡会很大程度上影响流水线的性能，需要编译器做出优化避免 RAW 相关，或者使用定向技术解决 RAW 相关。