山东大学＿＿计算机科学与技术＿＿＿学院

＿＿计算机体系结构＿＿＿课程实验报告

学号：202000130143	姓名：　郑凯饶	班级：　计科 1 班
实验题目：实验五　数据相关		
实验学时：2	实验日期：5.10	

实验目的：

通过本实验，加深对数据相关的理解，掌握如何使用定向技术来减少相关性带来的 stall。

硬件环境：

Dell Latitude 5411
Intel(R) Core(TM) i5-10400H CPU @ 2.60GHz(8GPUs),~2.6GHz

软件环境：

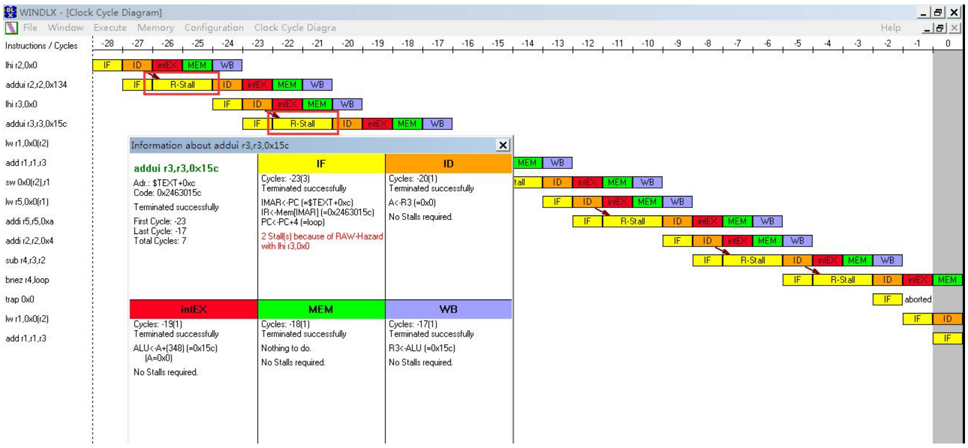
VMware Workstation 16 Player
Windows 7

实验步骤与内容：

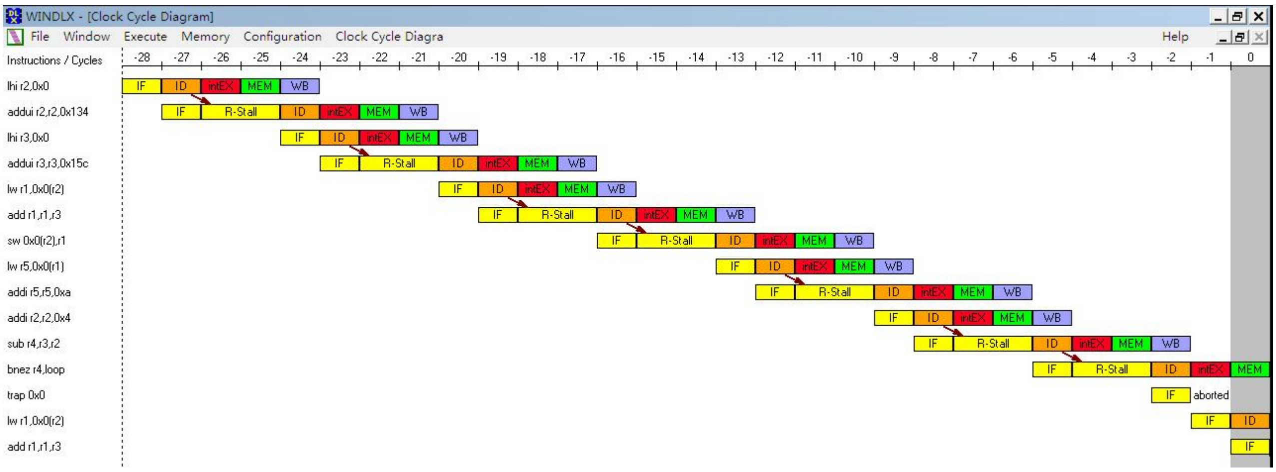
1. 阅读汇编代码，程序将 B 数组每个元素+10，并通过 A 数组保存 B 数组对应每个元素的内存地址：

```
        LHI     R2, (A>>16) & 0xFFFF
        ADDUI   R2, R2, A & 0xFFFF
        LHI     R3, (B>>16)&0xFFFF
        ADDUI   R3, R3, B&0xFFFF
        ; 将数组首地址加载至寄存器 R2,R3 中
    loop:
        LW      R1, 0 (R2)
        ADD     R1, R1, R3
        SW      0(R2), R1   ; 计算 B 数组各个元素的地址
        LW      R5, 0 (R1)
        ADDI    R5, R5, #10  ; 将 B 数组每个元素+10 但不保存
        ADDI    R2, R2, #4
        SUB     R4, R3, R2
        BNEZ    R4, loop
        TRAP    #0
    A: .word 0, 4, 8, 12, 16, 20, 24, 28, 32, 36
    B: .word 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
```

2. 关闭 Forwarding 选项，运行程序 data_d.s，分析数据相关：

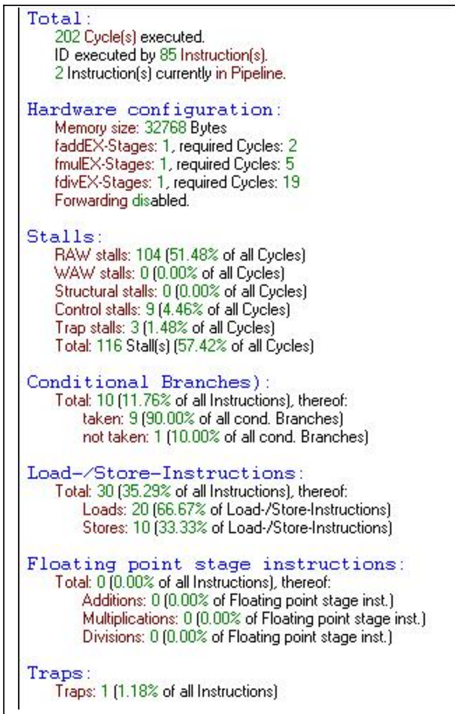


程序初始化过程中的两个 addui 指令发生数据相关，由于前面的 lhi 指令在 WB 阶段才将写回寄存器，这里可以理解为<u>前一条指令在 WB 前半周期写回寄存器，下一条指令在后半周期读取寄存器</u>。因此这两条指令均无法进行 ID，在 IF 部件停留两个 Stalls。



往后面分析，可以看到有规律地出现 R-stall，均是因为数据相关（前一条指令产生的结果，下一条指令使用到），箭头指示了相关的指令对，一次循环发生 4 次。

3. 打开 Statistics 界面，分析：


Total:
202 Cycle(s) executed.
ID executed by 85 Instruction(s).
2 Instruction(s) currently in Pipeline.

Hardware configuration:
Memory size: 32768 Bytes
faddEX-Stages: 1, required Cycles: 2
fmulEX-Stages: 1, required Cycles: 5
fdivEX-Stages: 1, required Cycles: 19
Forwarding disabled.

Stalls:
RAW stalls: 104 (51.48% of all Cycles)
WAW stalls: 0 (0.00% of all Cycles)
Structural stalls: 0 (0.00% of all Cycles)
Control stalls: 9 (4.46% of all Cycles)
Trap stalls: 3 (1.48% of all Cycles)
Total: 116 Stall(s) (57.42% of all Cycles)

Conditional Branches):
Total: 10 (11.76% of all Instructions), thereof:
taken: 9 (90.00% of all cond. Branches)
not taken: 1 (10.00% of all cond. Branches)

Load-/Store-Instructions:
Total: 30 (35.29% of all Instructions), thereof:
Loads: 20 (66.67% of Load-/Store-Instructions)
Stores: 10 (33.33% of Load-/Store-Instructions)

Floating point stage instructions:
Total: 0 (0.00% of all Instructions), thereof:
Additions: 0 (0.00% of Floating point stage inst.)
Multiplications: 0 (0.00% of Floating point stage inst.)
Divisions: 0 (0.00% of Floating point stage inst.)

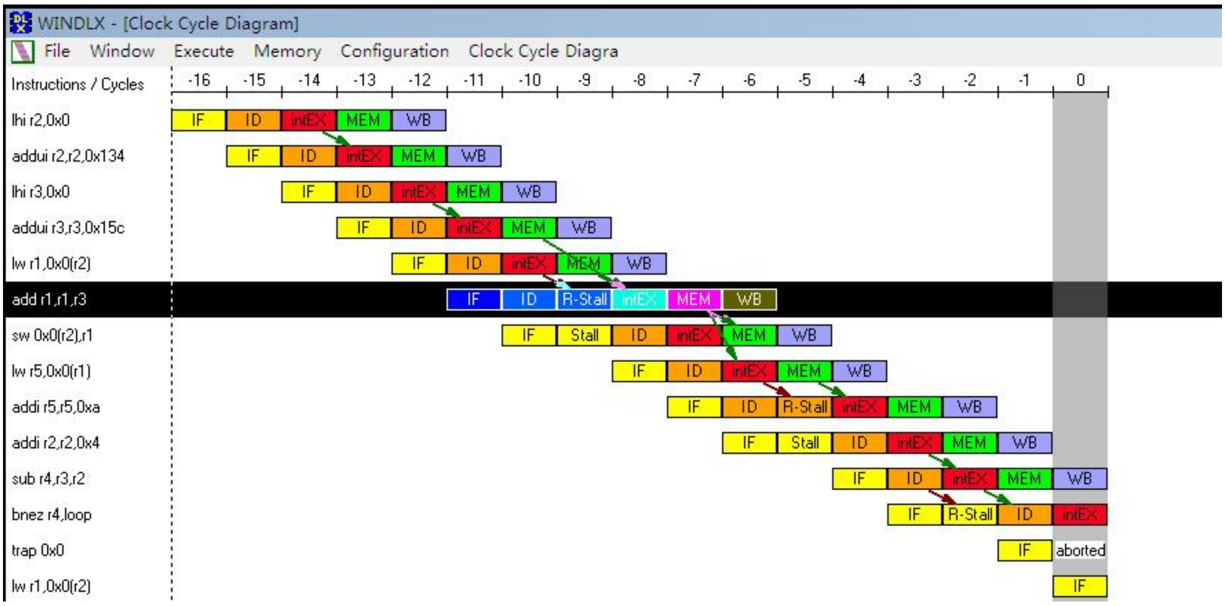
Traps:
Traps: 1 (1.18% of all Instructions)

数据相关引起的暂停时钟周期数：104（对应 RAW stalls，初始化有 4 个，一次循环产生 10 个，一共循环 10 次）

程序执行的总时钟周期数：202

暂停时钟周期数占总执行周期数的百分比：104/202=51.48%

4. 打开 Forwarding 选项，运行程序 data_d.s，分析数据相关以及定向技术的使用：



初始化的数据相关通过定向技术消除了，lui 指令的结果在 intEX 阶段计算得到之后定向到 addui 指令的 intEX 阶段，也就是送回 intEX 部件。

指令 lw r1,0x0[r2]和指令 add r1,r1,r3 的原有的数据相关消除了，add 指令不在 IF 阶段停留 2 个 stalls，但是新的相关产生了：lw 指令在 ID 周期读取寄存器的值，在 EX 周期计算访问地址，在 MEM 周期访存获取数据，因此后面的 add 指令需在 ID 停留 1 个 stall，等待 lw 指令的 MEM 周期完成。另外 add 指令通过定向得到寄存器 r3 的值。

5. 打开 Statistics 界面，分析：

```
Total:
    128 Cycle(s) executed.
    ID executed by 85 Instruction(s).
    2 Instruction(s) currently in Pipeline.

Hardware configuration:
    Memory size: 32768 Bytes
    faddEX-Stages: 1, required Cycles: 2
    fmulEX-Stages: 1, required Cycles: 5
    fdivEX-Stages: 1, required Cycles: 19
    Forwarding enabled.

Stalls:
    RAW stalls: 30 (23.44% of all Cycles), thereof:
        LD stalls: 20 (66.67% of RAW stalls)
        Branch/Jump stalls: 10 (33.33% of RAW stalls)
        Floating point stalls: 0 (0.00% of RAW stalls)
    WAW stalls: 0 (0.00% of all Cycles)
    Structural stalls: 0 (0.00% of all Cycles)
    Control stalls: 9 (7.03% of all Cycles)
    Trap stalls: 3 (2.34% of all Cycles)
    Total: 42 Stall(s) (32.81% of all Cycles)

Conditional Branches):
    Total: 10 (11.76% of all Instructions), thereof:
        taken: 9 (90.00% of all cond. Branches)
        not taken: 1 (10.00% of all cond. Branches)

Load-/Store-Instructions:
    Total: 30 (35.29% of all Instructions), thereof:
        Loads: 20 (66.67% of Load-/Store-Instructions)
        Stores: 10 (33.33% of Load-/Store-Instructions)

Floating point stage instructions:
    Total: 0 (0.00% of all Instructions), thereof:
        Additions: 0 (0.00% of Floating point stage inst.)
        Multiplications: 0 (0.00% of Floating point stage inst.)
        Divisions: 0 (0.00% of Floating point stage inst.)

Traps:
    Traps: 1 (1.18% of all Instructions)
```

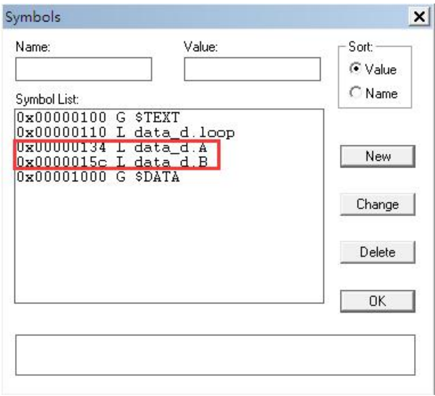
数据相关引起的暂停时钟周期数：30（对应 RAW stalls，一次循环产生 3 个，一共循环 10 次）

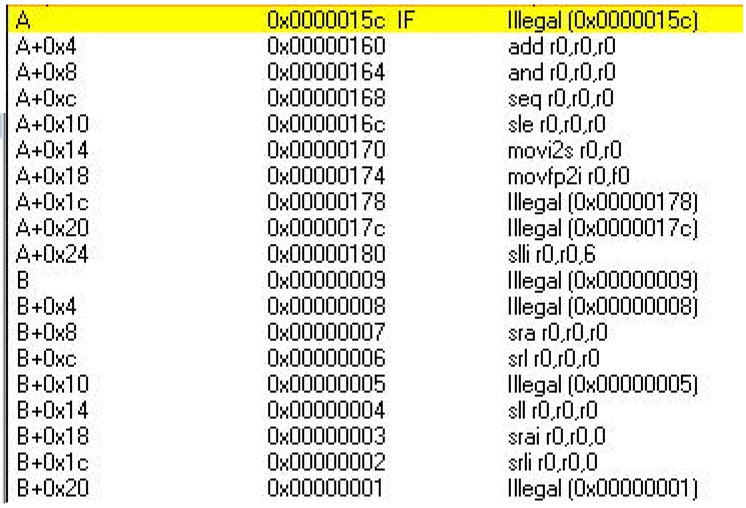
程序执行的总时钟周期数：128

暂停时钟周期数占总执行周期数的百分比：30/128=23.44%

定向技术减少了大量数据相关，流水线性能提升了 0.58 倍（202/128=1.58）。

6. 查看程序结果





可以看到数组 A 存储了数组 B 各个元素对应的内存地址，数组 B 的内容没有改变。

结论分析与体会：

这次实验，加深了我对流水线 5 个阶段的理解，理清了指令的执行细节。进一步体会到数据相关对流水线性能的影响，熟悉定向技术的基本应用。