5.12.2020

CS 224

Lab#4

Section 2

Zübeyir Bodur

21702382

1. Dissassembling in part b

```
        addi    $v0, $0, 5
        addi    $v1, $0, 12
        addi    $a3, $v1, -9
        or      $a0, $a3, $v0
        and     $a1, $v1, $a0
        add     $a1, $a1, $a0
        beq     $a1, $a3, L2
        slt     $a0, $v1, $a0
        beq     $a0, $0, L1
        addi    $a1, $0, 0
L1:     slt     $a0, $a3, $v0
        add     $a3, $a0, $a1
        sub     $a3, $a3, $v0
        sw      $a3, 68($v1)
        lw      $v0, 80($0)
        j       L2
        addi    $v0, $0, 1
L2:     sw      $v0, 84($0)
L3:     j       L3
```

2. RTL Descriptions for the new instructions

a) nop

IM[PC]
PC <== PC + 4

b) sw+

IM[PC]
DM[RF[rs] + SignExtend(imm)] <== RF[rt]
RF[rs] <== RF[rs] + 4
PC <== PC + 4

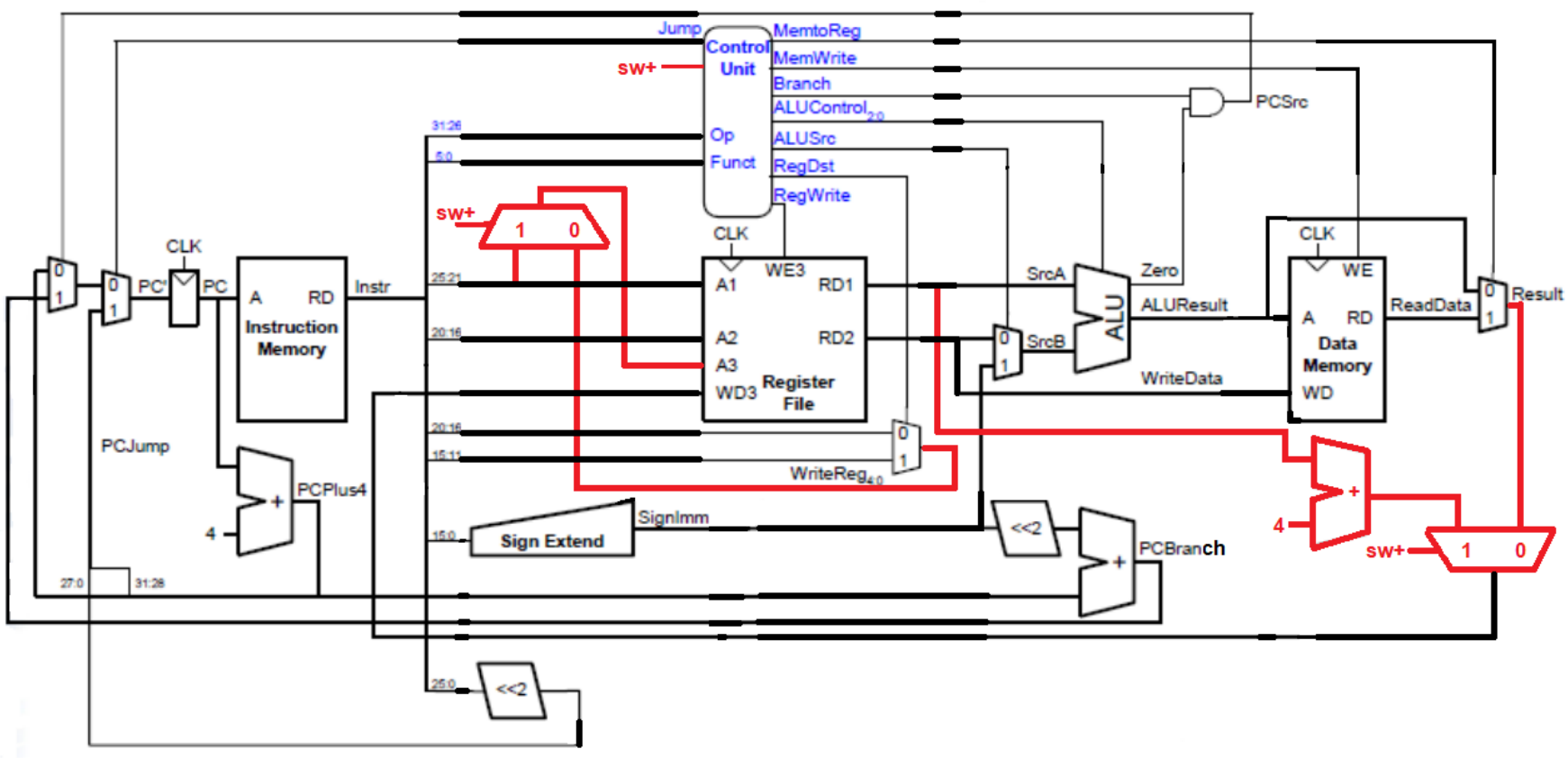
3. Changes made in the single cycle processor

For sw+, 2 main changes were made. One is for the write adress port (A3) of the register file, the other is for write data port (WD3) of the register file.

Before A3, a multiplexer is added to choose between $\text{WriteReg}_{3:0}$ (either rt or rd) and $\text{Instr}_{25:21}$ (rs) with a control signal “sw+”. If “sw+” is 1, $\text{Instr}_{25:21}$ will be chosen.

Before WD3, a multiplexer is added to choose between RD1 + 4 (meaning RF[rs] incremented by 4) and Result with the same control signal “sw+”. If “sw+” is 1, RD1 + 4 will be chosen. For this reason, an adder is also added to the datapath.

For nop, no changes were needed.


Jump
sw+
Control Unit
MemtoReg
MemWrite
Branch
ALUControl2:0
ALUSrc
RegDst
RegWrite
Op
Funct
31:26
5:0
PCSrc
sw+
1 0
CLK
PC'
PC
A
RD
Instruction Memory
Instr
25:21
20:16
15:11
15:0
25:0
A1
A2
A3
WD3
WE3
RD1
RD2
Register File
SrcA
SrcB
ALU
Zero
ALUResult
WriteData
CLK
WE
A
RD
Data Memory
WD
ReadData
Result
WriteReg4:0
PCJump
PCPlus4
4
Sign Extend
SignImm
<<2
PCBranch
27:0
31:28

4. Table for new instructions and control signals

sw+ column is added to the table as it's a new control signal.

Instr.	$Op_{5:0}$	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$	Jump	sw+
nop	111111	0	0	0	0	0	0	00	0	0
sw+	101111	1	0	1	0	1	0	00	0	1

5. Test MIPS Program

The following test program does iterative multiplication of 5 and 12, saves 60 into the memory and checks the result with the one in the memory. If result, $v0 is neither less than $t3 nor $t3 is less then $v0, we will have 0 in the result of the or operation ($t3).

Said program also assembles and runs succesfully in MARS, when sw+ is seperated into two instructions (sw and addi). nop is not deleted because it actually exists in MIPS instruction set, event thou it doesn't in Original10.

sw+ $a2, 4($t2) == > sw $a2, 4($t2) followed by addi $t2, $t2, 4

The following code is dissassembled operations for the test program in IMEM module of the design.

```
addi    $a0, $0, 5
addi    $s0, $0, 1
and     $t0, $a0, $0
addi    $a1, $0, 12
addi    $a2, $0, 68
addi    $t2, $0, 8
sub     $a3, $a2, $t2
addi    $t2, $t2, 0x10010000
sw+     $a2, 4($t2)
sw      $a3, 4($t2)
add     $v0, $0, $0
add     $t0, $a1, $0
slt     $t1, $0, $t0
beq     $t1, $s0, L1
```

```
        j       L2
        nop
L1:     add     $v0, $v0, $a0
        addi    $t0, $t0, -1
        slt     $t1, $0, $t0
        nop
        beq     $t1, $s0, L1
L2:     lw      $t3, 4($t2)
        slt     $t4, $t3, $v0
        slt     $t5, $v0, $t3
        or      $t3, $t4, $t5
```

And the following code is for testing purposes, to make sure that the logic of the code in IMEM is also working in MARS.

```
.text
addi $a0, $0, 5
addi $s0, $0, 1
and $t0, $a0, $0
addi $a1, $0, 12
addi $a2, $0, 68
addi $t2, $0, 8
sub $a3, $a2, $t2
addi $t2, $t2, 0x10010000
sw $a2, 4($t2)
addi $t2, $t2, 4
sw $a3, 4($t2)
add $v0, $0, $0
add $t0, $a1, $0
slt $t1, $0, $t0
beq $t1, $s0, L1
j L2
nop
L1: add $v0, $v0, $a0
addi $t0, $t0, -1
slt $t1, $0, $t0
nop
beq $t1, $s0, L1
L2: lw $t3, 4($t2)
slt $t4, $t3, $v0
slt $t5, $v0, $t3
or $t3, $t4, $t5
```

The contents of $v0 should be equal to the contents of the memory adress 0x10010010, and $t3 should finally have 0.