Assignment 3

Rachel Hwang

October 24, 2013

1. 4.1.1 A
   RegDst = 1
   ALUSrc = 0
   MemtoReg = 0
   RegWrite = 1
   MemRead = 0
   MemWrite = 0
   Branch = 0
   ALOp1 = 1
   ALOp2 = 0

2. 4.1.2 A
   a. **Program Counter** supplies instruction address to instruction memory.
   b. **Adder** adds 4 to the PC.
   c. **Instruction memory** specifies register operands.
   d. **Mux** gets value from adder to update PC.
   e. **Instruction memory** specifies register operands.
   f. **Mux** determines that the second opreand is from a register.
   g. **Registers** supply specified operands.
   h. **ALU** performs the AND operation.
   i. **Mux** detetmines that value passed back to registers comes from ALU.
   j. **Registers** are updated with values from ALU.
   So every block is used except the branch add block and D-Mem.

3. 4.1.3 A
   The second adder produces an output which is not used update the PC (controlled by the Mux). The ALU produces a 1-bit signal which is not used since this is not a branch instruction. The Data memory produces no output at all.

4. 4.1.4 A
   MIPS AND Critial Path:
   $I - mem \rightarrow Regs \rightarrow Mux \rightarrow ALU \rightarrow Mux \rightarrow Regs$
   $200ps + 90ps + 20ps + 90ps + 20ps + 90ps = 510ps$

5. 4.1.5 A
   MIPS LW Critical Path: $I - mem \rightarrow Regs \rightarrow Mux \rightarrow ALU \rightarrow D - mem \rightarrow Mux \rightarrow Regs$
   $200ps + 90ps + 20ps + 90ps + 250ps + 20ps + 90ps = 760ps$

6. 4.1.6 A
   MIPS BEQ Critical Path: $I - mem \rightarrow Regs \rightarrow Mux \rightarrow ALU \rightarrow Mux$
   $200ps + 90ps + 20ps + 90ps + 20ps = 420ps$

7. 4.7.1 A
   MIPS (ADD, AND, etc.) Critial Path:
   $I - mem \rightarrow Mux \rightarrow Regs \rightarrow Mux \rightarrow ALU \rightarrow Regs$
   $200ps + 20ps + 90ps + 20ps + 90ps + 90ps = 510ps$

8. 4.7.2 A
   MIPS (LW) Critial Path:
   $I - mem \rightarrow Regs \rightarrow ALU \rightarrow D - Mem \rightarrow Regs$
   $200ps + 90ps + 90ps + 250ps + 90ps = 720ps$

9. 4.7.3 A
   MIPS (ADD, BEQ, LW and SW) Critial Path:
   $I - mem \rightarrow Mux \rightarrow Regs \rightarrow Mux \rightarrow ALU \rightarrow D - Mem \rightarrow Mux \rightarrow Regs$
   $200ps + 20ps + 90ps + 20ps + 90ps + 250ps + 20ps + 90ps = 780ps$

10. 4.7.4 A
    Only in LW and SW = 35% of all cycles.

11. 4.7.5 A
    The sign-extend unit in only needed when working with immediate values, so in ADDI, BEQ, LW and SW. This totals to 80% of all cycles. When this input is not needed, the sign-extend unit still takes the upper 16 bits of the instruction and passed the extended result to the Mux anyway, it is just not used.

12. 4.7.6 A
    ADD/ADDI Critial Path: $I - mem \rightarrow Regs \rightarrow Mux \rightarrow ALU \rightarrow Mux \rightarrow Regs$
    $Time = 200ps + 90ps + 20ps + 90ps + 90ps = 490ps$

    BEQ Critial Path: $I - mem \rightarrow Regs \rightarrow Mux \rightarrow ALU \rightarrow Mux$
    $Time = 200ps + 90ps + 20ps + 90ps + 20ps = 420ps$

    LW Critial Path: $I - mem \rightarrow Mux \rightarrow Regs \rightarrow Mux \rightarrow ALU \rightarrow D - Mem \rightarrow Mux \rightarrow Regs$
    $200ps + 20ps + 90ps + 20ps + 90ps + 250ps + 20ps + 90ps = 780ps$

SW Critial Path: $I - mem \rightarrow Regs \rightarrow Mux \rightarrow ALU \rightarrow D - Mem$
$Time = 200ps + 90ps + 20ps + 90ps + 250ps = 650ps$

Since the critical path of our cycle is determined by LW, the instruction that takes the longest, we should improve the component that takes the longest in that path, which is the D-Mem component. Improving D-Mem by 10% means that the improved time for LW $= 200ps + 90ps + 20ps + 90ps + 225ps + 20ps + 90ps = 735ps$
$Speed - up = 25ps, (.1 \cdot 250ps)/780ps = 3.2\%$

13. 4.9.1 B
    If R1 = 1, R2 = 2, and R3 = 3 then value SLT(R1, R2, R3) =
    000000 00010 00011 00001 00000 101010
    OP..... RS.... RT.... RD... SHT.. FUNC..

14. 4.9.2 B
    The register supplied to "Read register 1" is R2. The register supplied to "Read register 2" is R3. Both registers are actually read.

15. 4.9.3 A
    The register supplied to "Write register" is R1. This register is actually written.

16. 4.9.4 A
    Jump = 0
    RegDst = 1

17. 4.9.5 A
    Op codes:
    j.. = 000010
    lw. = 100011
    sw. = 101011
    beq = 000100
    add = 000000

Input						Output
Op1	Op2	Op3	Op4	Op5	Op6	Jump
X	X	X	X	1	0	1
Otherwise						0

18. 4.9.6 A

Input						Output	
Op1	Op2	Op3	Op4	Op5	Op6	Jump	RegDst
X	X	X	X	1	0	1	0
X	X	X	X	0	X	0	1
X	X	X	X	1	1	0	0

19. 4.11.1 B
    000000 00100 00010 00001 00000 101010
    Sign-extend (15-0) $\rightarrow$ 00000000000000000000100000101010 = 0x0000082A
    Shift left 2 (25-0) $\rightarrow$ 0x20820A8
    This represents SLT R1, R4, R2.

20. 4.11.2 B
    ALU control input = 0111
    ALUSrc = 0
    ALOp1 = 1
    ALOp2 = 0

21. 4.11.3 B
    new PC = PC+4
    $PC \rightarrow Add \rightarrow Mux \rightarrow Mux$

22. 4.11.4 B
    RegDst Mux: 1 because this is an R-type instruction, so outputs R1 as the value of write register.
    ALUSrc Mux: 0 because this uses rt as its second operand, so outputs R2 = -128.
    MemtoReg Mux: 0 because this is not an SW/LW instruction, so output is the ALU result = 0.
    Branch Mux: 0 because this is not a branch instruction, so outputs PC+4
    Jump Mux: 0 because this is not a jump instruction, so outputs PC+4

23. 4.11.5 B
    For the ALU, data input values are -32(R4) and -128(R2).
    For the first add unit, input values are PC and 4.
    For the second add unit, input values are PC+4 and 00 0000 0000 0000 0000 0010 0000 1010 1000 = 0x0000020A8.

24. 4.11.6 B
    Inputs for the registers unit:
    Read register 1: R4
    Read register 2: R2
    Write register: R1
    Write data: 0
    Reg Write: 0

25. 7.14.2 B
    for (i=0; i ¡ 2000; i++)

for (j=0; j¡3000; j++)
X_array[i][j] = Y_array[j][i] + 200;

Let base address of X_array be stored in $s0, Y_array be stored in $s1.

8-wide SIMD
I'm out of time.