CS 224

Lab No. 4

Section: 1

Ömer Oktay Gültekin

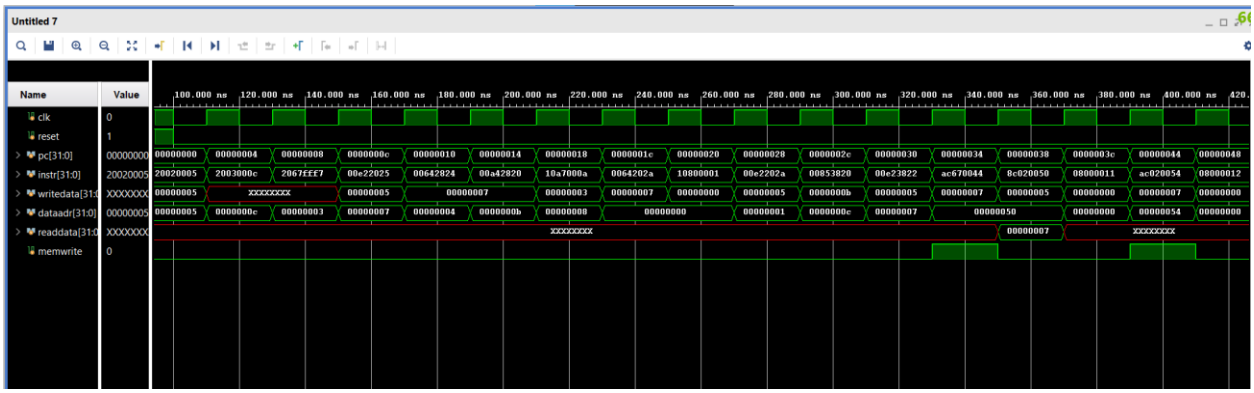
Bilkent ID: 21901413

Part 1:

a)

Location (Hex)	Machine Instruction (Hex)	Assembly Language Equivalent
00	20020005	addi $v0, $zero, 5
04	2003000c	addi $v1, $zero, 12
08	2067fff7	addi $a3, $v1, -9
0c	00e22025	or $a0, $a3, $v0
10	00642824	and $a1, $v1, $a0
14	00a42820	add $a1, $a1, $a0
18	10a7000a	beq $a1, $a3, 0x0000000a
1c	0064202a	slt $a0, $v1, $a0
20	10800001	beq $a0, $zero, 0x00000001
24	20050000	addi $a1, $zero, 0
28	00e2202a	slt $a0, $a3, $v0
2c	00853820	add $a3, $a0, $a1
30	00e23822	sub $a3, $a3, $v0
34	ac670044	sw $a3, 68($v1)
38	8c020050	lw $v0, 80($zero)
3c	08000011	j 0x00000044
40	20020001	addi $v0, $zero, 1
44	ac020054	sw $v0, 84($zero)
48	08000012	j 0x00000048

d)



e)

i) writedata corresponds to second operand that is rt register in assembly language format.

ii) in regfile module, rd1 and rd2 assigned at all times whereas, wd3 wrote in regfile only at positive edge of clock. Therefore, when writedata has contents of $v1 and $a3 registers, they haven't written yet. Thus, we see them as undefined. However, if we look at them after one clock cycle, we can see their values initialized.

iii) in dmem module, readdata always assigned to the content of the adress (ALUResult) of the RAM. However, since we do not store any words in early stages of the program, it always reads uninitialized words but when we store $a3 in address 80 and read address 80 we get a value.

iV) In an R-type instruction dataadr correspond to the ALUResult that is the result of the operation controlled by ALUControl where operands are rs and rt registers (not immediate).

V) memwrite becomes 1 for sw instructions.

f)

```
module alu(input  logic [31:0] a, b,
           input  logic [2:0]  alucont,
           output logic [31:0] result,
           output logic zero);

   always_comb
      case(alucont)
         3'b010: result = a + b;
         3'b011: result = a << b;
         3'b110: result = a - b;
         3'b000: result = a & b;
         3'b001: result = a | b;
         3'b111: result = (a < b) ? 1 : 0;
         default: result = {32{1'bx}};
      endcase

   assign zero = (result == 0) ? 1'b1 : 1'b0;
endmodule
```

Part 2:

a) sracc RTL expression:

IM[PC]

RF[rd] ← RF[rs] << RF[rt] + RF[rd]

PC ← PC + 4

sw+ RTL expression

IM[PC]

DM[RF[rs] + SignExt(immed)]←RF[rt]

RF[rs] ← RF[rs] + 4

PC ← PC + 4

b)

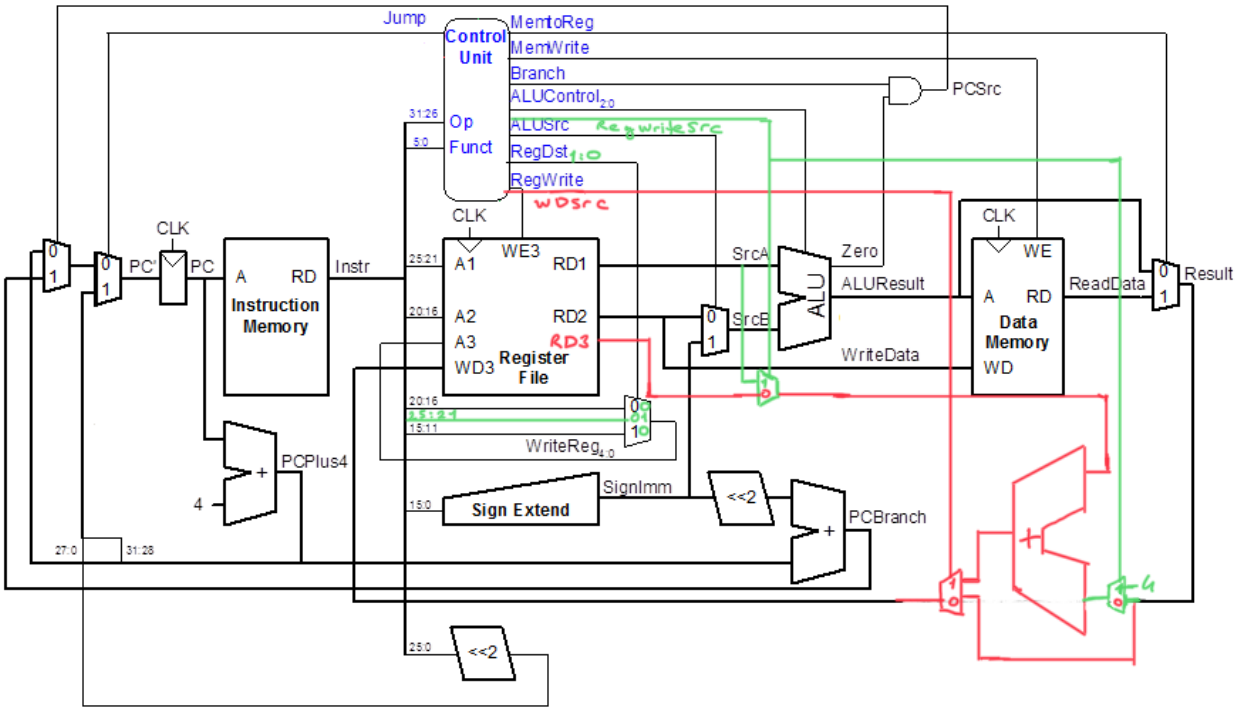


Figure 1: Datapath for Extended12

c)

Instruction	Opcode	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp	Jump	WDSrc	RegWriteSrc
R-type	000000	1	10	0	0	0	0	10	0	0	X
lw	100011	1	00	1	0	0	1	00	0	0	X
sw	101011	0	XX	1	0	1	X	00	0	0	X
beq	000100	0	XX	0	1	0	X	01	0	0	X
addi	001000	1	00	1	0	0	0	00	0	0	X
j	000010	0	XX	X	X	0	X	XX	1	0	X
sracc	111110	1	10	0	0	0	0	1X	0	1	0
sw+	111111	1	01	1	0	1	X	00	0	1	1

Table 1: Main Decoder for Extended12

ALUOp	Funct	ALUControl
00	X	010 (add)
01	X	110 (subtract)
1X	100000 (add)	010 (add)
1X	100010 (sub)	110 (subtract)
1X	100100 (and)	000 (and)
1X	100101 (or)	001 (or)
1X	101010 (slt)	111 (set less than)
1X	111111 (sra)	011 (shift right arithmetic)

Table 2: ALU Decoder for Extended12