CO224 – Computer Architecture

Lab 5 Part 5– Building a Simple Processor

Group 11 :- Mapagedara T.L.B (E/20/248), Yogesh R.J (E/20/453)

Newly added instructions:

`bne <offset> <r1> <r2>`: Branch not equal=> if r1,r2 not equal take branch given by offset

`mult <r3> <r1> <r2>` : Multiplies the values in r2 and r3 and stores result in r3

`sll <r3> <r2> <offset>`: Logically left shifts r2 by offset value and stores result in r3

`srl <r3> <r2> <offset>`: Logically right shifts r2 by offset value and stores result in r3

`sra <r3> <r2> <offset>`: Arithmetically right shifts r2 by offset value and stores result in r3

`ror <r3> <r2> <offset>`: Rotates r2 right by r3 amounts

BNE is implemented by making slight difference to BEQ instruction handle module.

Those other instruction have separate units implement in ALU to perform the operations.

Table 1: OPCODEs for Instruction Set

Instruction	**OPCODE**
loadi	00000000
mov	00000001
add	00000010
sub	00000011
and	00000100
or	00000101
j	00000110
beq	00000111
bne	00001000
mult	00001001
sll	00001010
srl	00001011
sra	00001100
ror	00001101

Table 2: ALU Functions

ALUOP	Function	Operations	Supported Instructions	Unit's Delay
000	FORWARD	DATA2→RESULT	loadi, mov	#1
001	ADD	DATA1+DATA2→RESULT	add, sub, beq, bne	#2
010	AND	DATA1&DATA2→RESULT	and	#1
011	OR	DATA1 | DATA2→RESULT	or	#1
100	LSHIFT	DATA1<<DATA2→RESULT	sll	#2
101	RSHIFT	• DATA1>>DATA2→RESULT • DATA1>>>DATA2→RESULT • Rotate Right DATA1 by DATA2 amounts	srl, sra, ror	#2
110	MULT	DATA1 x DATA2→RESULT	mult	#3

CPU Datapath

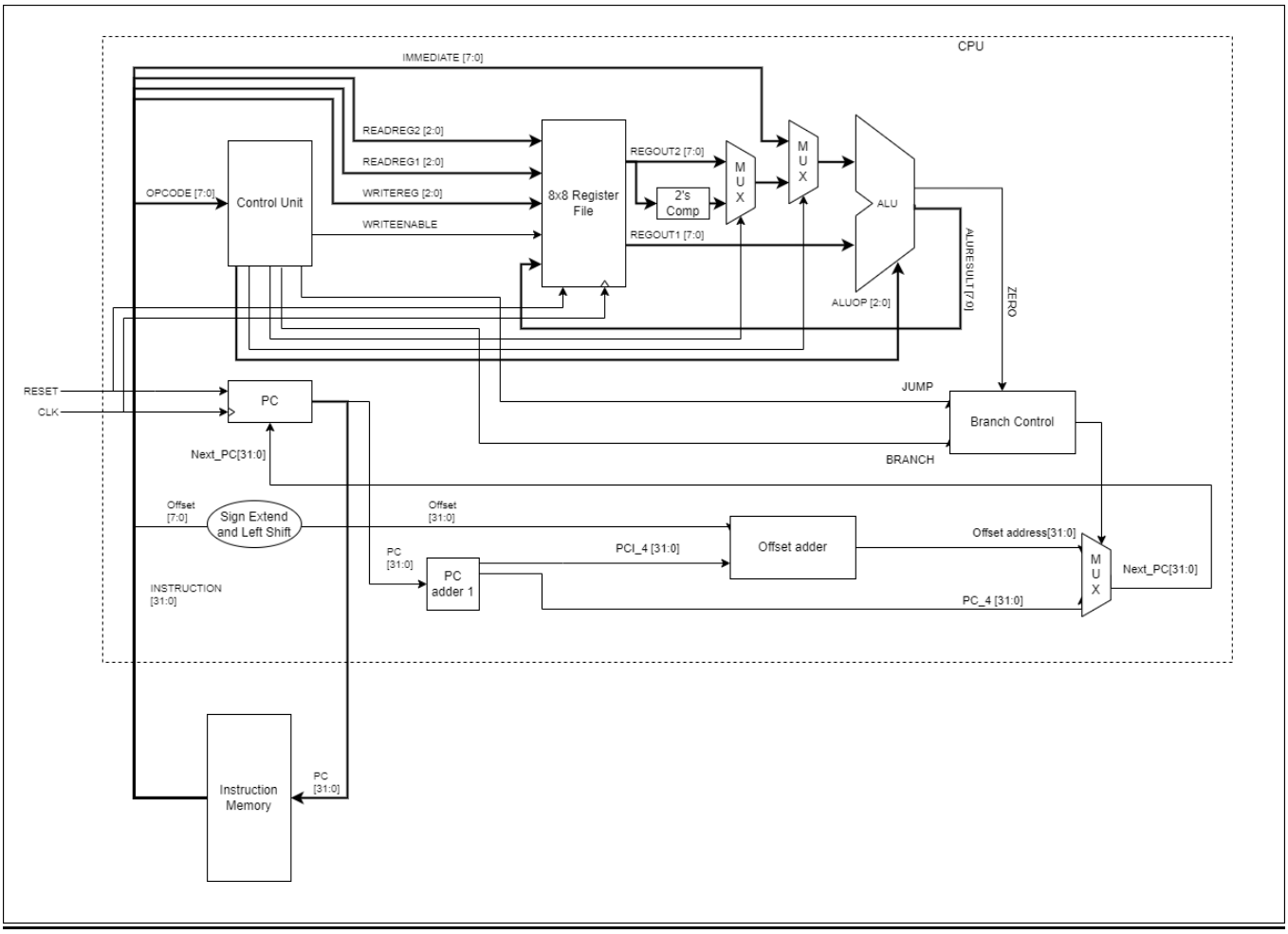

Figure 1: CPU Block Diagram

BNE Instruction

Branch and jump instructions (BEQ,BNE,J) are controlled by Branch Control module as shown in figure1. Control Unit making Branch and Jump signal appropriately to given instruction.

Table 3: Branch Control Module Behavior

Branch	Jump	Branch Select
0	0	0
0	1	1
1	0	1 if Zero is HIGH
1	1	1 if Zero is LOW

Branch_Select = jump ^ (branch & zero)

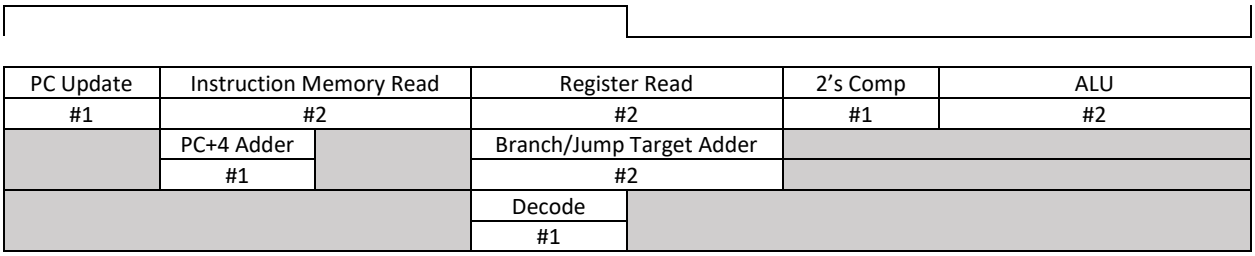

Figure 2: Timing details for BNE instruction

MULT Instruction

Separate unit added for Multiplication consist of array of Full adders. Full adders was implemented using basic combinational unit. (XOR,AND,OR)

Since the MULT unit uses a large array of Full Adders simulation delay of #3 time units was added to this unit.

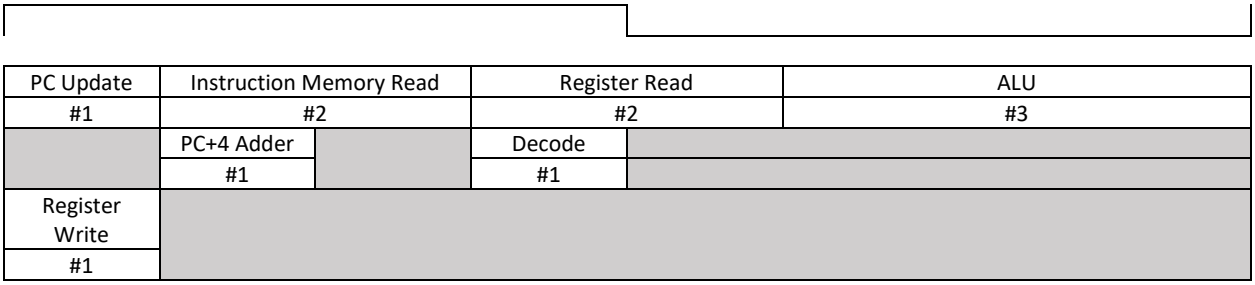

Figure 3: Timing details for MULT instruction

SLL Instruction

Perform bitwise left shift operation. Seperate unit was added to the ALU. This unit acts as a barrel shifter with several layers of Muxes to generate the shifted output.

Since the Shift amount is 8 bits, the shifter supports shift operations only up to 8 shifts.

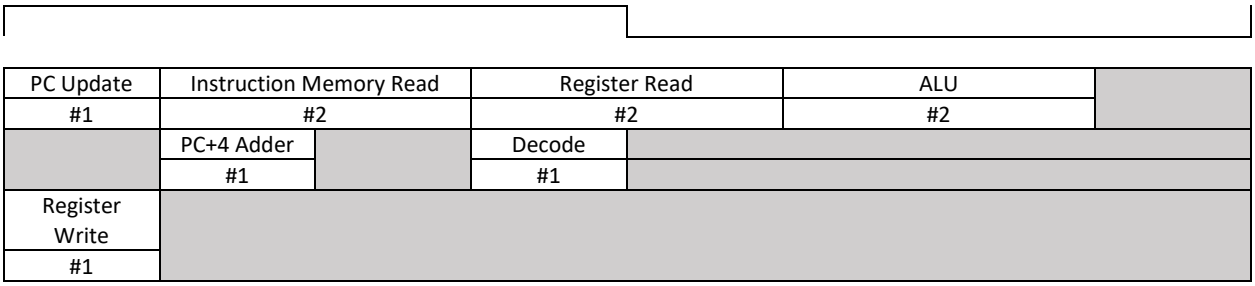

Figure 4: Timing details for SLL instruction datapath

RSHIFT Unit

This unit handles three operations.

1. Logical right shift (srl)
2. Arithmetic right shift (sra)
3. Rotate right (ror)

This unit chooses the correct operation to perform based on the first two MSBs of the shift value provided in DATA2.

Table 3: Behaviour of RSHIFT Unit

DATA2[7:6]	Operation Performed
00	srl
01	sra
10	ror

This unit also uses layers of muxes to perform shifting operation.

For the rotate right instruction output repeats its pattern every 8 shifts. So, the higher order bits of large shift values can simply be ignored and the result would still be same.

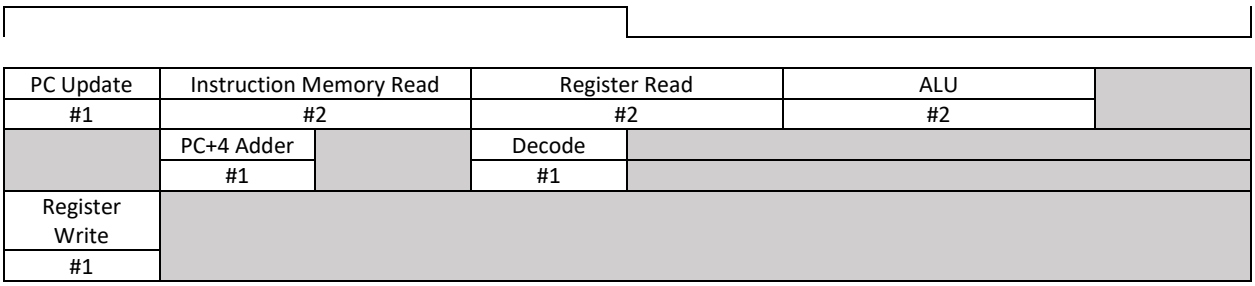

Figure 5: Timing details for right shift instruction