CO224-Computer Architecture

Extended ISA -LAB 5 – part 5

Group 19

E/20/100 – Fernando A.I.

E/20/280 – Pathirage R.S.

In part 5, We have extended to new seven instructions.

The newly added instructions as well as their expected usage is as follows.

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

sll : Logically left shifts r2 by offset value and stores result in r1

srl : Logically right shifts r2 by offset value and stores result in r1

sra : Arithmetically right shifts r2 by offset value and stores result in r1

sla : Arithmetically left shifts r2 by offset value and stores result in r1

ror : Rotates r2 right by offset value and stores result in r1

rol : Rotates r2 left by offset value and stores result in r1

bne : Branches based on offset if values in r1 and r2 are not equal

To facilitate these new instructions, a minor modification was made to the Branch/Jump hardware and some additional functional units were implemented within the ALU of the CPU. These changes are reflected in the CPU block diagram and the ALU functions table given below.

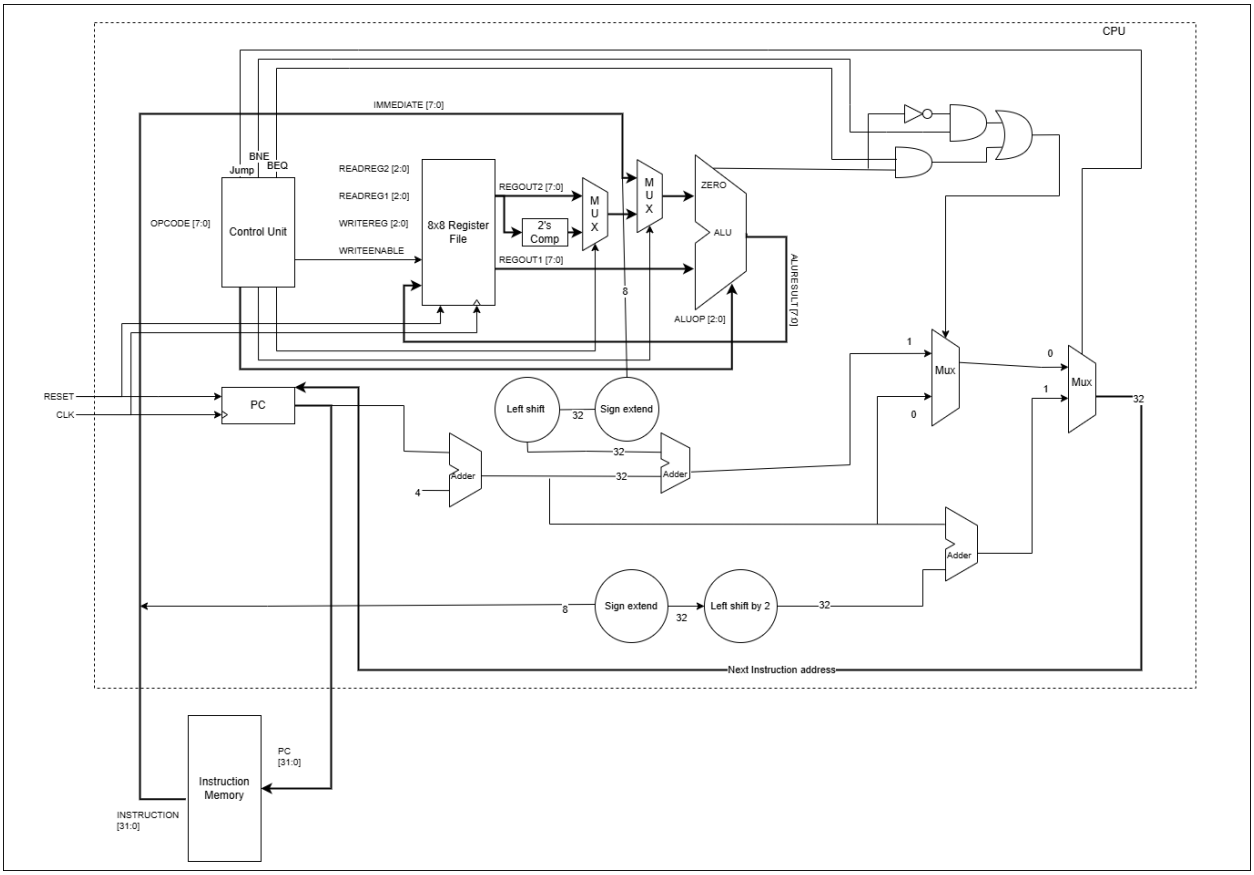

Figure 1: CPU Block Diagram

ALU

Table 1: ALU Functions

Select	Function	instructions	Delay
000	forward	loadi, mov	#1
001	add	add, sub, beq, bne	#2
010	and	and	#1
011	or	or	#1
100	mult	mult	#3
101	lshift	sll, srl	#2
110	ashift	sra, sla	#2
111	rotation	ror, rol	#2

Mult Instruction

For the MULT instruction, a separate functional unit was added to the ALU. The unit consists of an array of Full Adders that work in several layers to generate the result for each bit of the output. The Full Adder was also implemented as a module using basic combinational logic. As such, the block diagram for the MULT functional unit is as follows.

Multiplication of two 8-bit numbers a product of 16 bits. Since output is restricted to 8 bits, only 8 bits were calculated by the circuit. Also, timing diagram for Mult instruction is added.

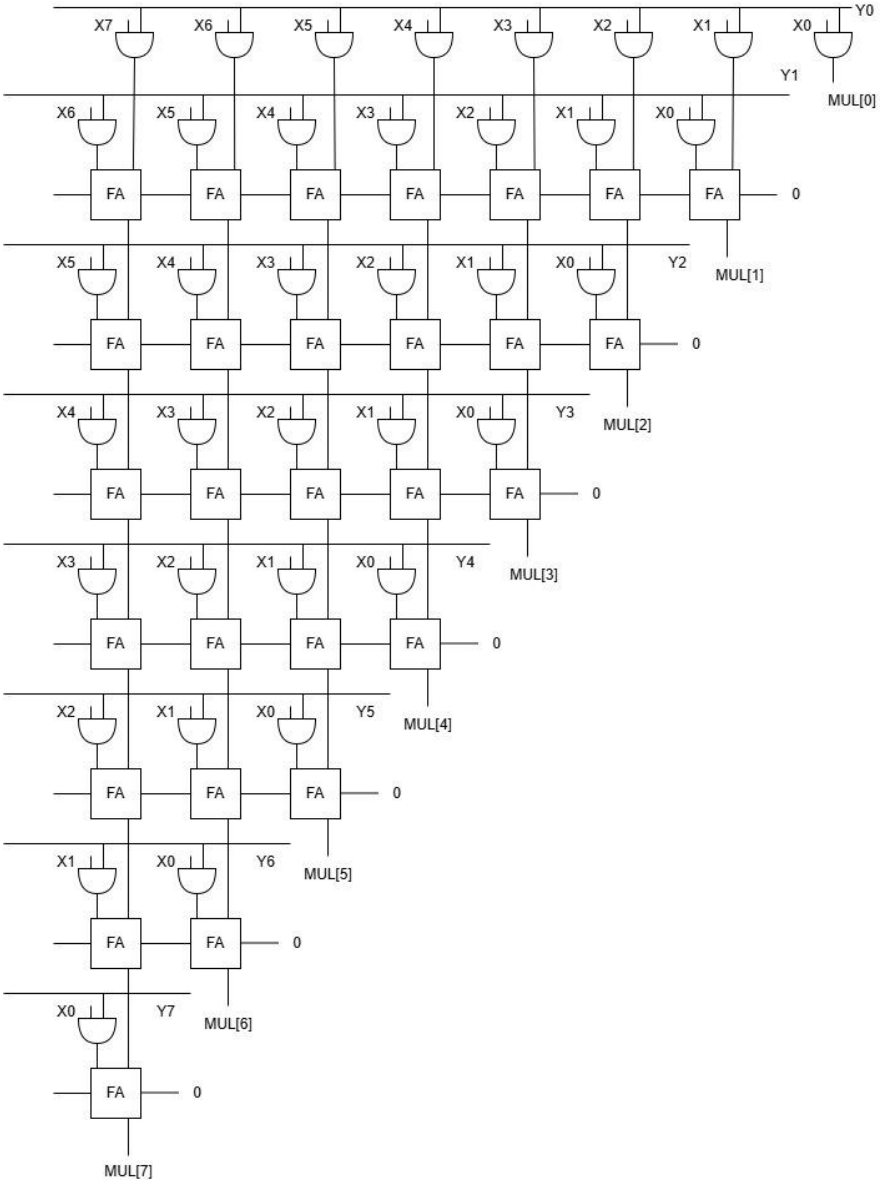

Figure 2: MULT Functional Unit Block Diagram

PC Update	Instruction Memory Read		Register Read	ALU
#1	#2		#2	#3
	PC+4 Adder		Decode	
	#1		#1	
Register Write				
#1				

Figure 3 : Timing details for MULT instruction data path

Logical Shift Instruction

Another functional unit was added to the ALU to facilitate bitwise Logical shift operations. This unit acts as a barrel shifter with several layers of MUXes to generate the shifted output. The MUXes for the unit were also implemented as separate modules and the block diagram for the LSHIFT functional unit is given below. Also, timing diagram for Mult instruction is added.

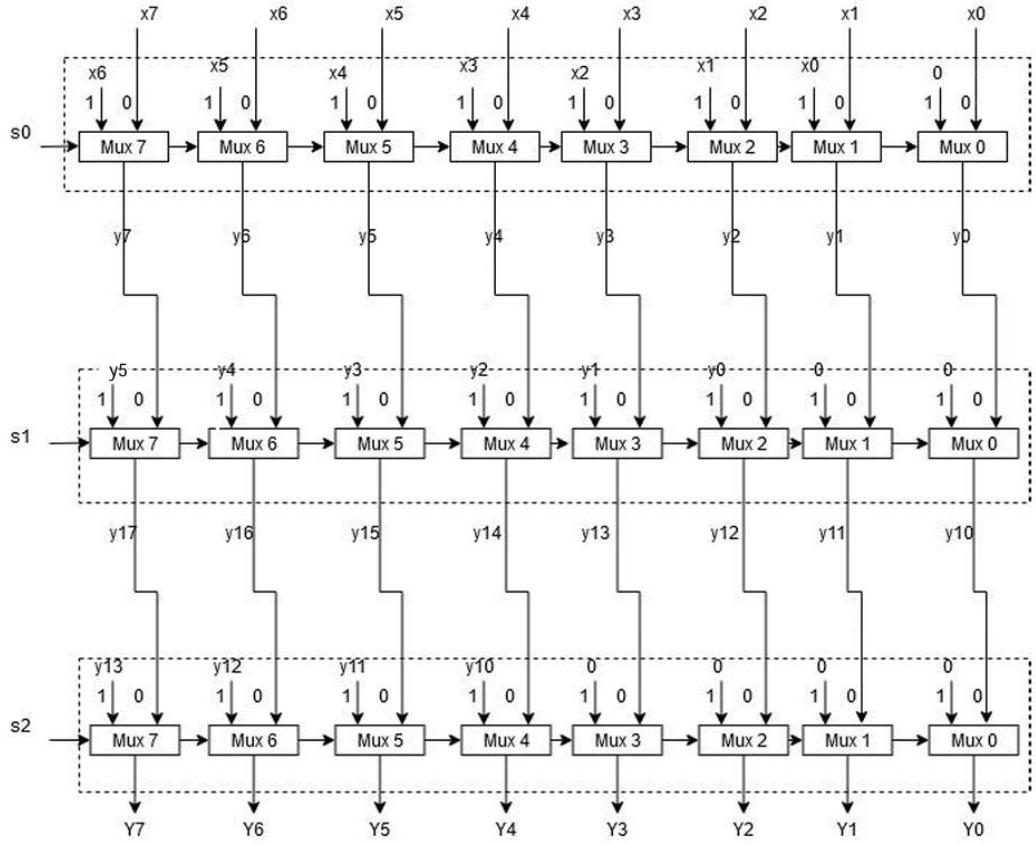

Figure 4: Logical left shift Unit Block Diagram

PC Update	Instruction Memory Read		Register Read	ALU	
#1	#2		#2	#2	
	PC+4 Adder		Decode		
	#1		#1		
Register Write					
#1					

Figure 5: Timing details for sll instruction data path

Arithmetic Shift Instruction

Another functional unit was added to the ALU to facilitate bitwise arithmetic shift operations. This also acts as barrel shift method. The MUXes for the unit were also implemented as separate modules and the block diagram for the LSHIFT functional unit is given below. Also, timing diagram for Mult instruction is added.

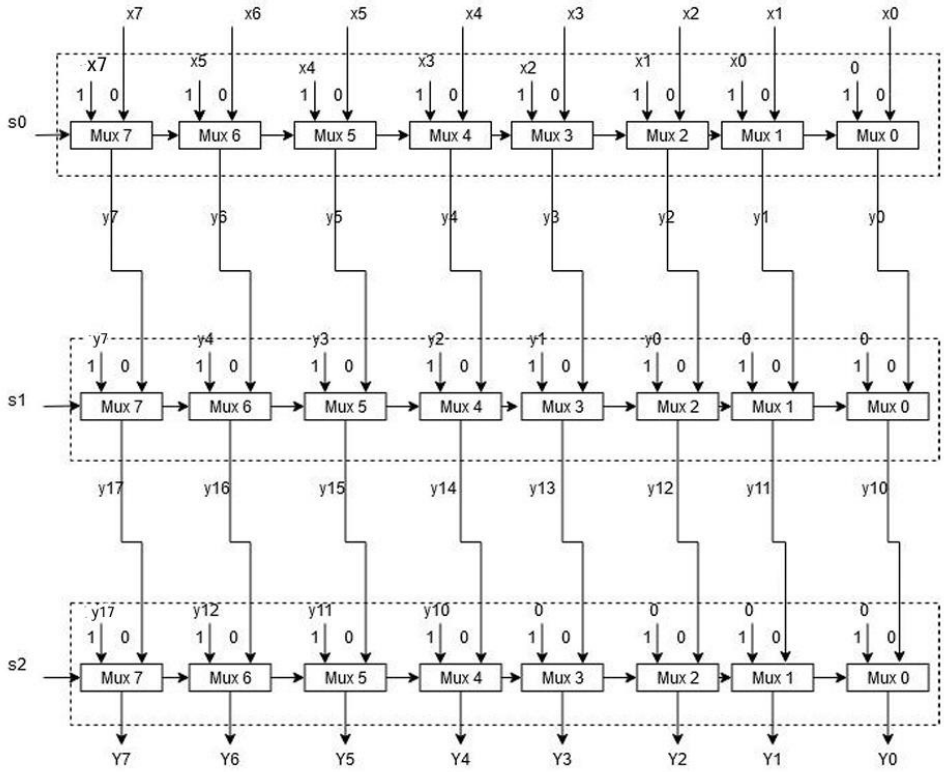

Figure 6: Arithmetic left shift Unit Block Diagram

PC Update	Instruction Memory Read		Register Read	ALU	
#1	#2		#2	#2	
	PC+4 Adder		Decode		
	#1		#1		
Register Write					
#1					

Figure 7: Timing details for sla instruction data path

Rotational Shift Instruction

This also acts as barrel shift method.

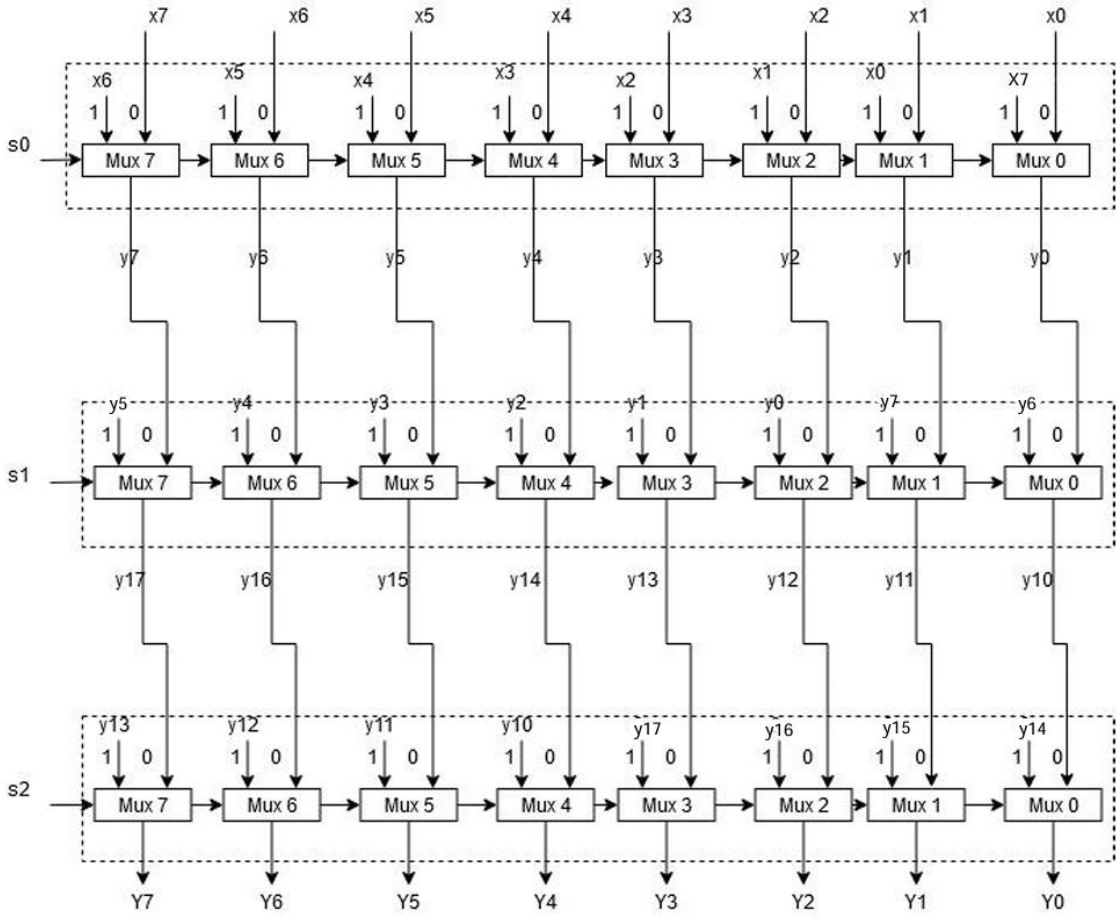

Figure 8: Rotational left shift Unit Block Diagram

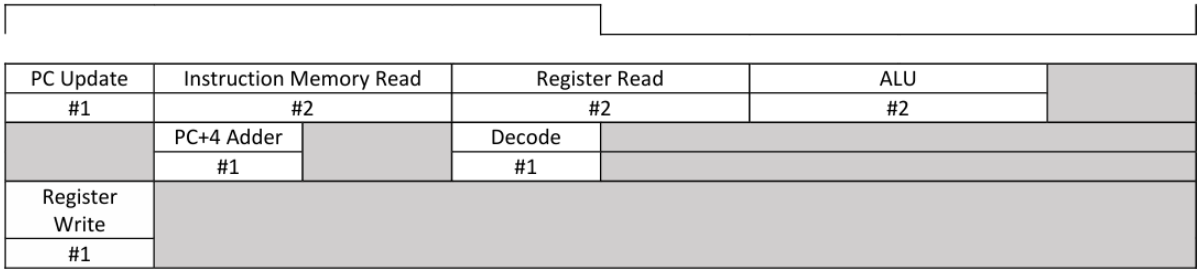

Figure 9: Timing details for rol instruction data path

BNE Instruction

As can be seen in the CPU block diagram (figure 1), the Flow Control Unit has been modified by replacing the AND gate , Not gate and OR gate.

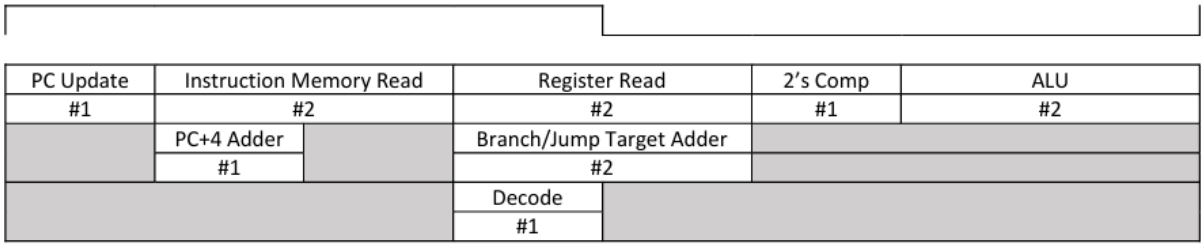

Figure 10: Timing details for BNE instruction data path

CPU OPCODES

Table 2: CPU OPCODEs for Instruction Set

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

Sample Commands

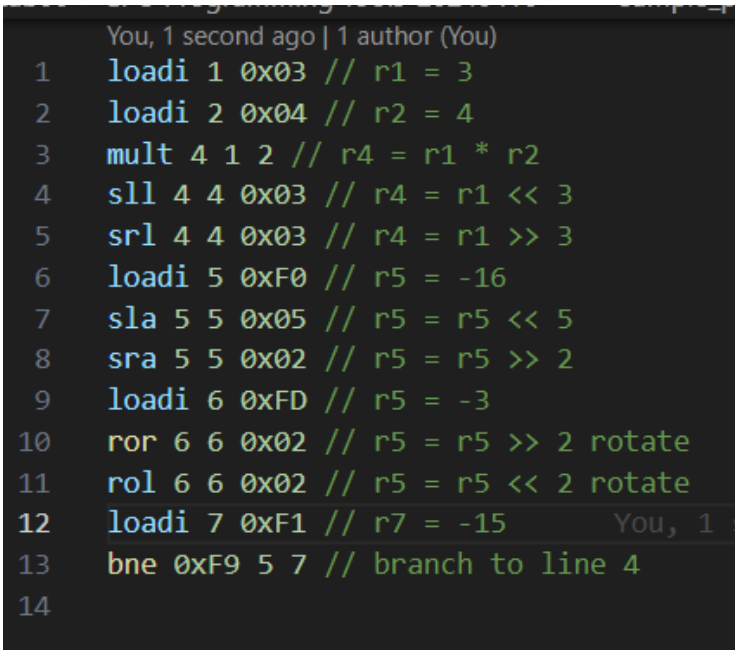

Figure 11: Commands

GTKWAVE Diagrams for above instructions

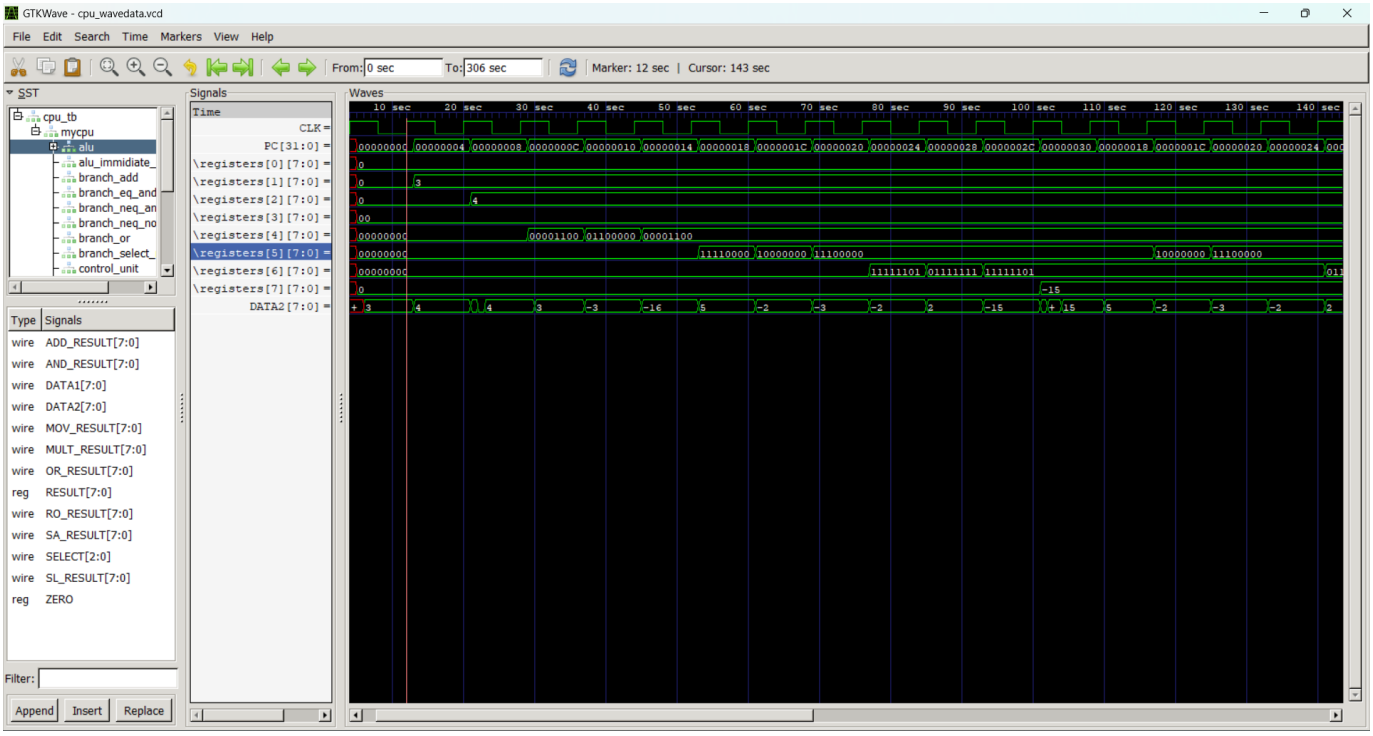

Figure 12: GTKwave Diagrams