Bahri Batuhan BİLECEN 2374635

EE446 LAB 4 & 5 Preliminary Work

Custom 16-bit ISA

- This custom 16-bit ISA is inspired by the 32-bit ARM instruction set.
- Suitable hardware for this ISA contains:
  - Von-Neumann architecture memory of 64 words, each word is of W=8 bits (byte-addressable memory)
    - 6 bits are required for addressing the memory
  - 8 general purpose registers of size 8 bits
    - 3 bits are required for addressing the general-purpose registers
  - PC (R7) and LR (R6) of size 8 bits and are inside the general-purpose registers

	Unified Data and Instruction Memory		
		0xC4	Some data
…			
…		0xB2	Instr #1
0x02		0x01	
0x01		0x03	Instr #0
0x00		0x71	

R7=PC	0x00
R6=LR	0x06
R5	0x05
R4	0x04
R3	0x03
R2	0x02
R1	0x01
R0	0x00

OP = 00 (Arithmetic operations)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP		CMD		I	Rd										
00		CMD		0	Rd			Rn			X	X	Rm		
00		CMD		1	Rd			rot			imm5				

CMD	function	pseudo instruction

00	ADD	Rd ← Rn CMD Rm if I = 0
		Rd ← Rd CMD (extimm5 ROL rot) if I = 1

01	SUB	

10	CMP	X ← Rn SUB Rm if I = 0
		X ← Rd SUB (extimm5 ROL rot) if I = 1

11	MOV	Rd ← (extimm5 ROL rot) and I = 1

Example possible instructions (Represented in ARM assembly mnemonic)
CMD R5,R3,R0
CMD R4,#20 (imm5=20, rot=0)
CMD R2,#80 (imm5=20, rot=2)

OP = 01 (Logic operations and shifting)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP		CMD			Rd										
01		CMD			Rd			Rn			X	X	Rm		
01		CMD			Rd			rot			imm5				

CMD	function	pseudo instruction
000	AND	Rd ← Rn CMD Rm
001	OR	
010	XOR	
011	ROL	Rd ← Rd CMD (extimm5)
100	ROR	
101	LSL	
110	LSR	
111	ASR	

- 3-bit shamt (0-7) can move the extended 5-bit immediate value to all possible locations in the 8-bit register. ROR is handled by ROL. (ROR(x) = ROL(8-x))

Example possible instructions (Represented in ARM assembly mnemonic)
AND R5,R3,R0
LSL R4, #2 (rot = X, imm5 = 2)
ROL R3, #6 ROR R3, #2 (imm5 = 8-2 = 6)

OP = 10 (Memory operations)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP		L	I	H	Rd			Rn			addr5/offset5				

L	function	pseudo instruction

1	LDR	Rd ← MEM[Rn + {H, offset5}] if I = 0
		Rd ← MEM[{H, addr5}] if I = 1

0	STR	Rd → MEM[Rn + {H, offset5}] if I = 0 Rd → MEM[{H, addr5}] if I = 1

- H determines which half of the memory to be addressed. Normally, addr/offset bits should take 6 bits. However, to keep the bit positions of Rn & Rd in the same place for all instruction types, immediate value is considered to have 5 bits. Most significant bit is H.
- H is handled in extender as the most significant bit.

Example possible instructions (Represented in ARM assembly mnemonic)
LDR R0, [R1,#3]
LDR R0, #63 (offset5=31, H=1)
STR R1, [R0]
STR R0, #2

OP = 11 (Branching)

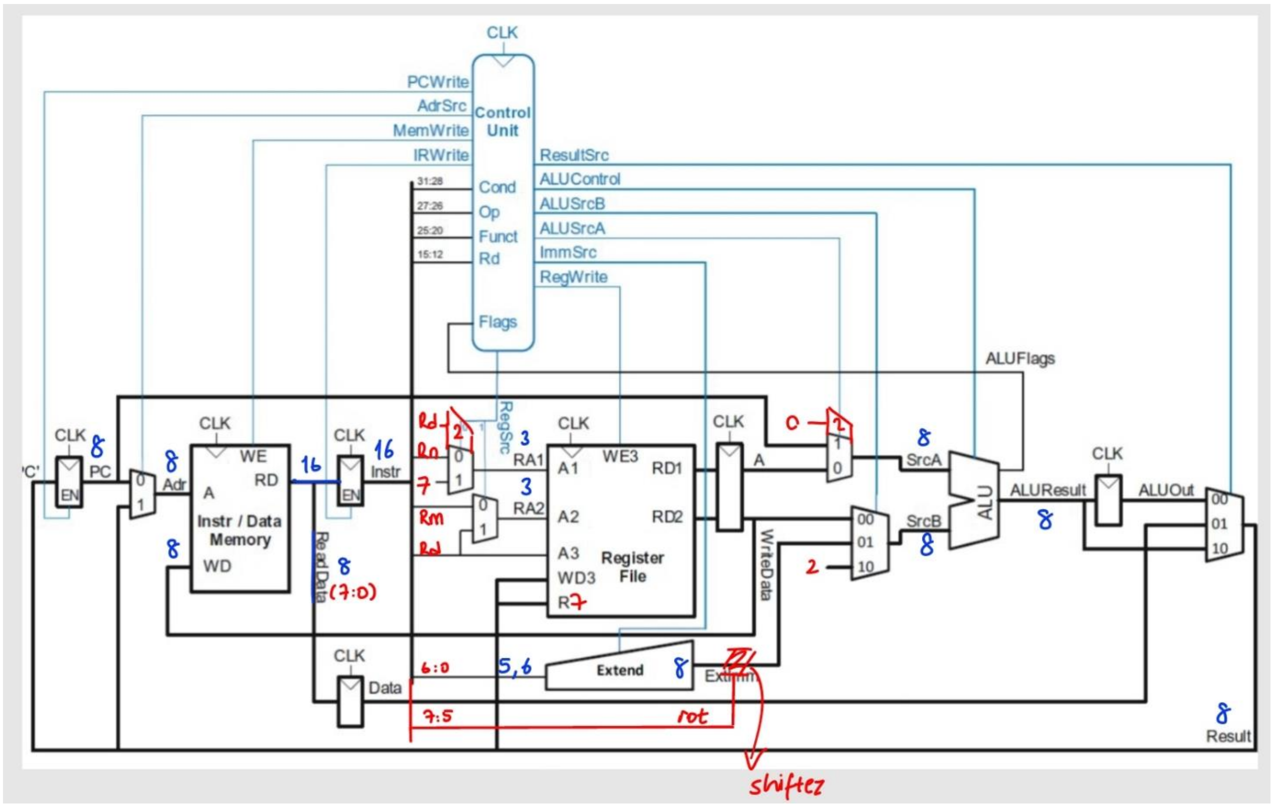
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP		type		flag		X	X	X	X	addr6					
11		00		XX		X	X	X	X	addr6					
11		10		XX		X	X	X	X	addr6					
11		01		X	1	1	0	X	X	addr6					

type	flag	function	explanation	pseudo instruction
00	X	B	branch	PC + 2 ← addr6
01	X	BL	branch with link	LR(R6) ← PC PC + 2 ← addr6
10	X	BI	branch indirect	PC + 2 ← MEM[addr6]
11	00	BEQ	branch if zero	B if condition satisfied
11	01	BNE	branch if not zero	
11	10	BHS/BCS	branch if carry	
11	11	BLO/BCC	branch if not carry	

- Each instruction is 2 bytes. The memory is byte addressable. Due to the architecture's 3-stage pipeline, branch target address is loaded to PC + (2 instructions ahead) = PC + (2*1) = PC + (2).
- Rd=R6 needs to be set for BL, hence Instr[10:8] = 3'b110.

Datapath Design

- Harris & Harris' multicycle ARM datapath design is modified.



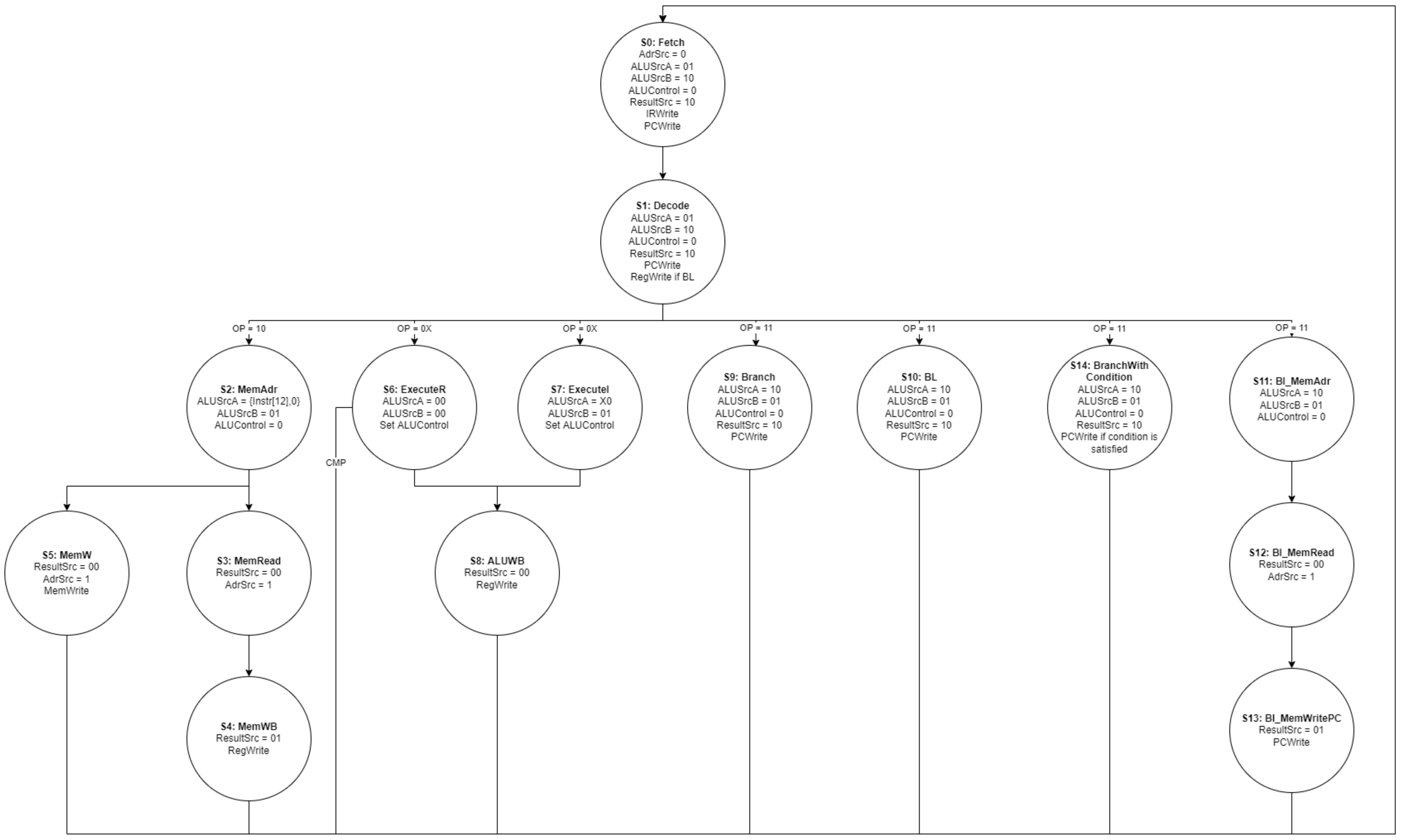
New datapath. Blue numbers show the datapath length.

- Datapath length is changed to accommodate 16-bit ISA. Instruction length is 16 bits and data length is 8 bits.
- RegSrc0 MUX is extended to accommodate new inputs. Register file's R7 input and RegSrc0[1]=immediate 7 are not used, however.
- For shifting and rotations, a shifter is placed after the extender. Rotation amount is Instr[7:5]=rot.
- To sum with 0, ALUSrcA is extended with immediate 0 input.
- ALU is modified to handle shifting and rotating operations inherently.
- For PC+(2 instructions), ALUSrcB[10] = immediate 2.

Controller Design

- An FSM in Verilog is written for the controller.
- It is worthy to note that some states can be combined or eliminated completely, since this state machine is not minimal.

Bahri Batuhan BİLECEN 2374635



Subroutines for Validation of Operation of ISA, Datapath and Controller

- Verifying all ISA operations are done within below subroutines. No additional tests are done for verifying all possible operations explicitly.

1. **Test of BI and BL**

- Initially, Rn = #n and PC = 0.

Address	Instruction	Explanation
0	STR R2,M[20]	M[20] = 2
2	ADD R0,R0,1	
4	CMP R0,2	
6	BEQ 10	
8	BI 20	B M[20]=2
10	BL 0	B 0, R6(LR)🡐12
12	Next instruction	

2. **2's complement of K**

- Initially, Rn = #n and PC = 0.
- 2's complement of K = ~K + 1 = K XOR 11111111 + 1 = K XOR (0-1) + 1
- R1 is the result.

Address	Instruction
0	SUB R0,R0,1
2	MOV R1,K
4	XOR R1,R1,R0
6	ADD R1,R1,1

3. **Sum of an array of 5 elements**

- Initially, Rn = #n and PC = 0.
- baseAdr = 26, offset=0, R0=0. Store 0🡒M[26], 2🡒M[27], 4🡒M[28], 6🡒M[29], 8🡒M[30]. When R0=10 and R1=31, break the loop. Load written data back into R2 and accumulate to R3. When R1=0, break the loop. R3 is the sum of 5 elements.

Address	Instruction
0	MOV R1, offset
2	MOV R3, 0
4	STR R0, M[R1+baseAdr]
6	ADD R1,R1,1
8	ADD R0,R0,2
10	CMP R0,10
12	BNE 2
14	SUB R1,R1,1
16	LDR R2, M[R1+baseAdr]
18	ADD R3,R2,R3
20	CMP R1,0
22	BNE 14
24	DONE

4. **Evenness/oddity detection**

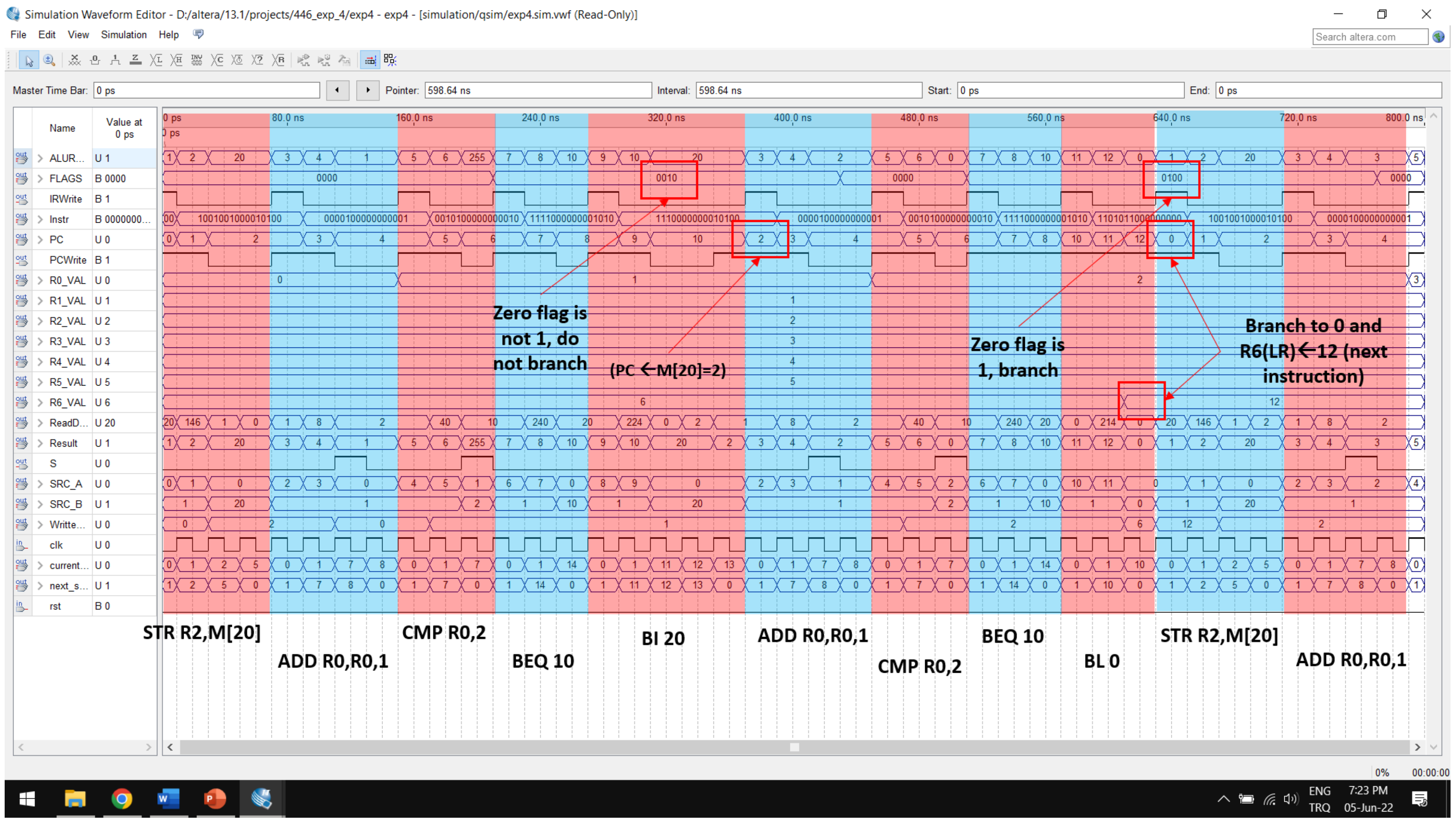
- Initially, Rn = #n, PC = 0, number = $n_7n_6n_5n_4n_3n_2n_1n_0$.

Address	Instruction	Explanation
0	MOV R0, number	R0: $n_7n_6n_5n_4n_3n_2n_1n_0$
2	MOV R2, number	R2: $n_7n_6n_5n_4n_3n_2n_1n_0$
4	AND R3,R1,R0	R3: $0000000n_0$
6	SUB R2,R2,R3	R2: $n_7n_6n_5n_4n_3n_2n_1 0$
8	CMP R3,0	
10	BNE 20	If branches, odd
12	MOV R3, 30	R3: 00011110
14	AND R2,R2,R3	R2: $000n_4n_3n_2n_1 0$
16	SHL R2,2	R2: $0n_4n_3n_2n_1 000$
18	B 22	
20	ROL R2,3	R2: $n_4n_3n_2n_1 0n_7n_6n_5$
22	DONE	

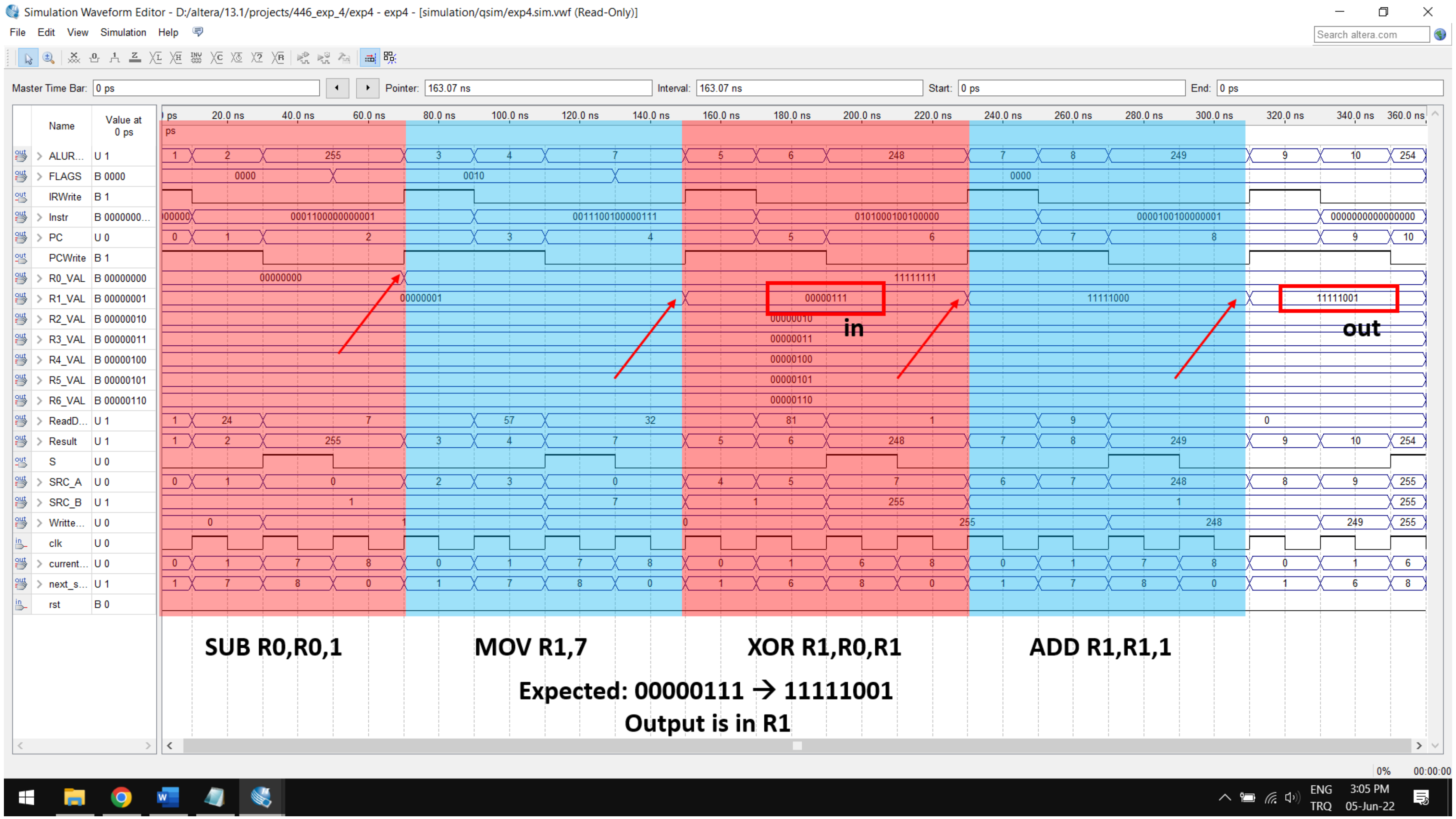
Simulation Tests

- Tests are done with waveforms instead of ModelSim due to ModelSim causing a lot of performance issues on my personal computer.

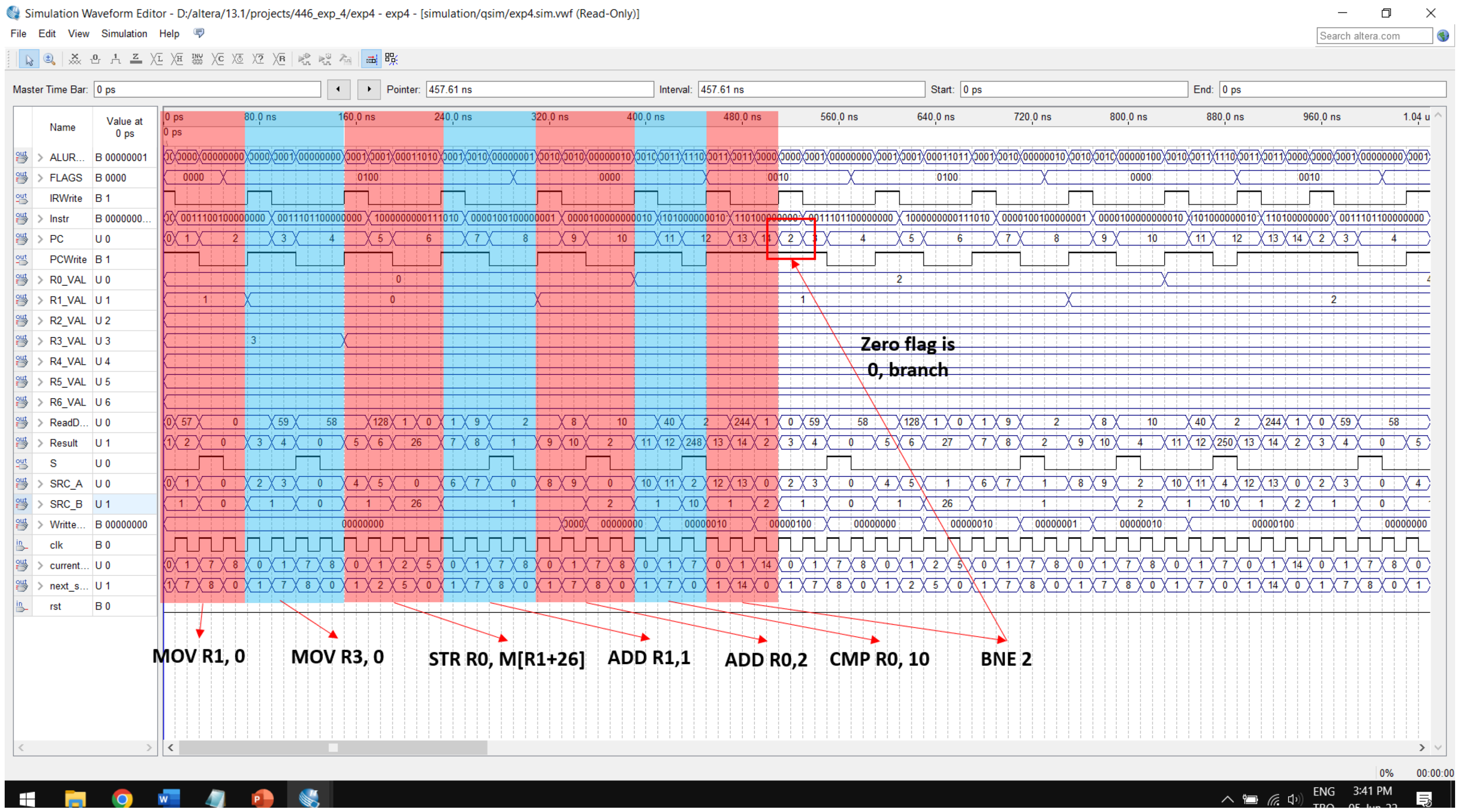
Bahri Batuhan BİLECEN 2374635



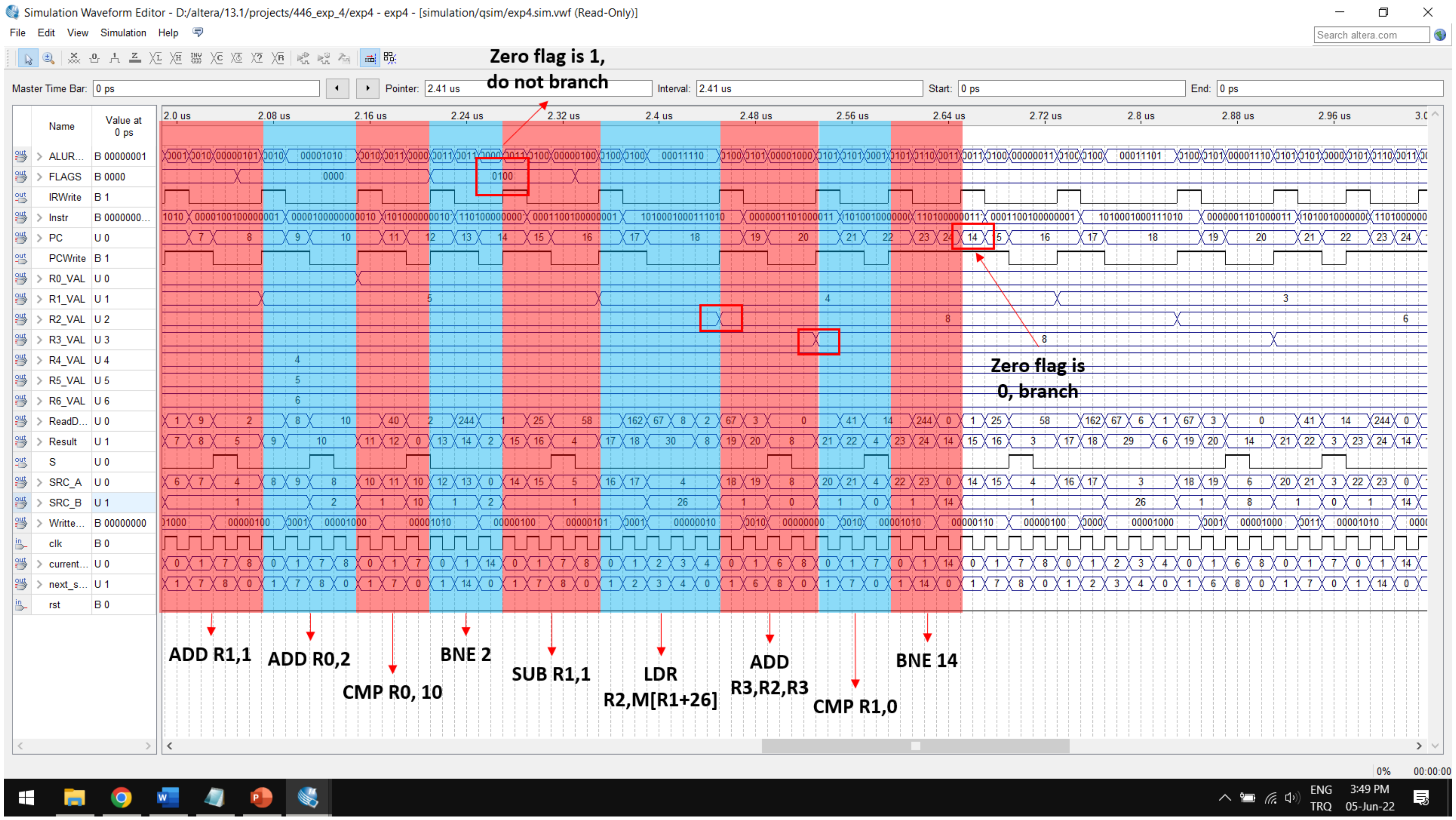
Test of BI and BL.


Simulation Waveform Editor - D:/altera/13.1/projects/446_exp_4/exp4 - exp4 - [simulation/qsim/exp4.sim.vwf (Read-Only)]
in
out
SUB R0,R0,1
MOV R1,7
XOR R1,R0,R1
ADD R1,R1,1
Expected: 00000111 → 11111001
Output is in R1

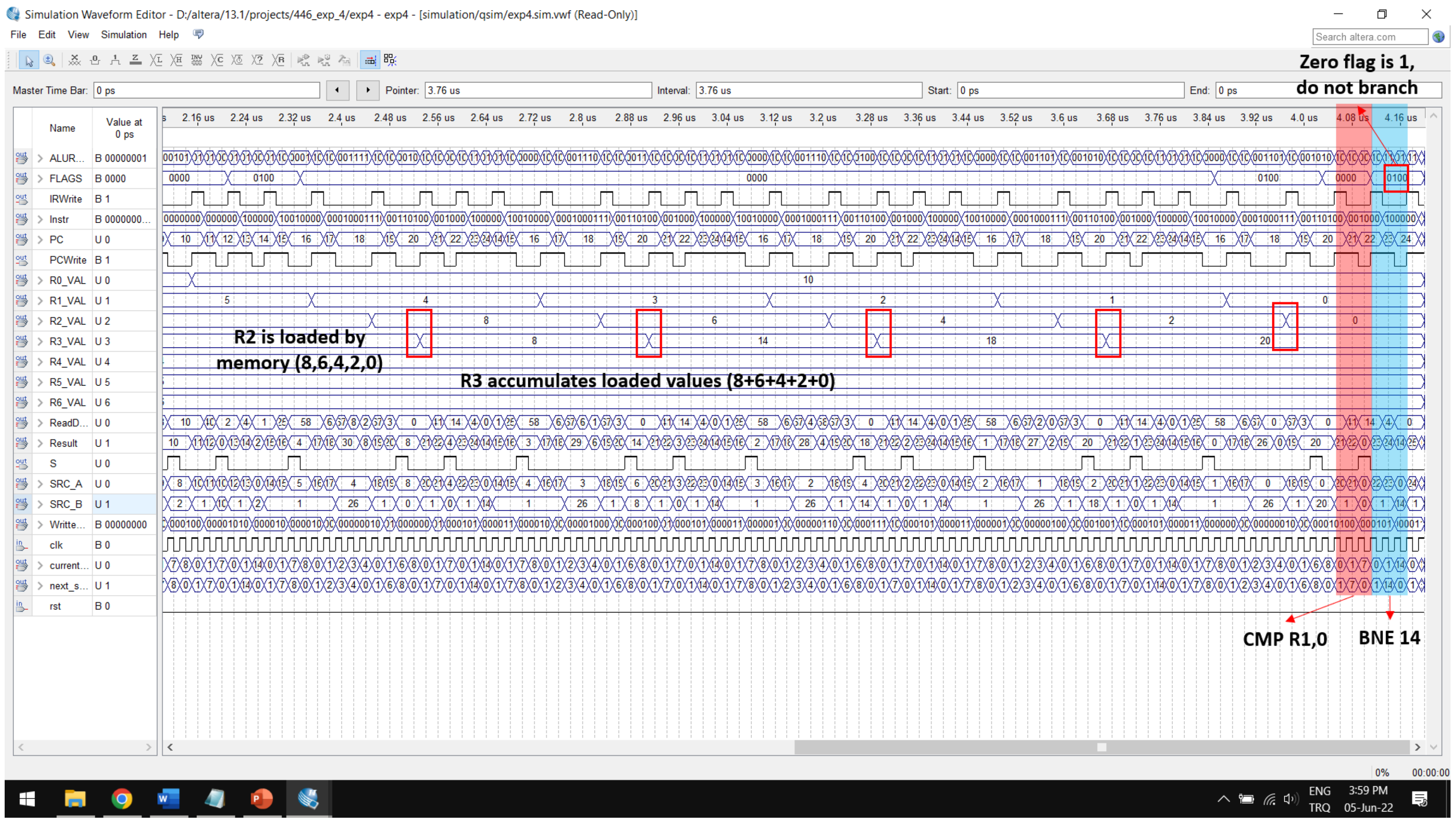
2's complement operation.


Simulation Waveform Editor - D:/altera/13.1/projects/446_exp_4/exp4 - exp4 - [simulation/qsim/exp4.sim.vwf (Read-Only)]
Zero flag is 0, branch
MOV R1, 0
MOV R3, 0
STR R0, M[R1+26]
ADD R1,1
ADD R0,2
CMP R0, 10
BNE 2

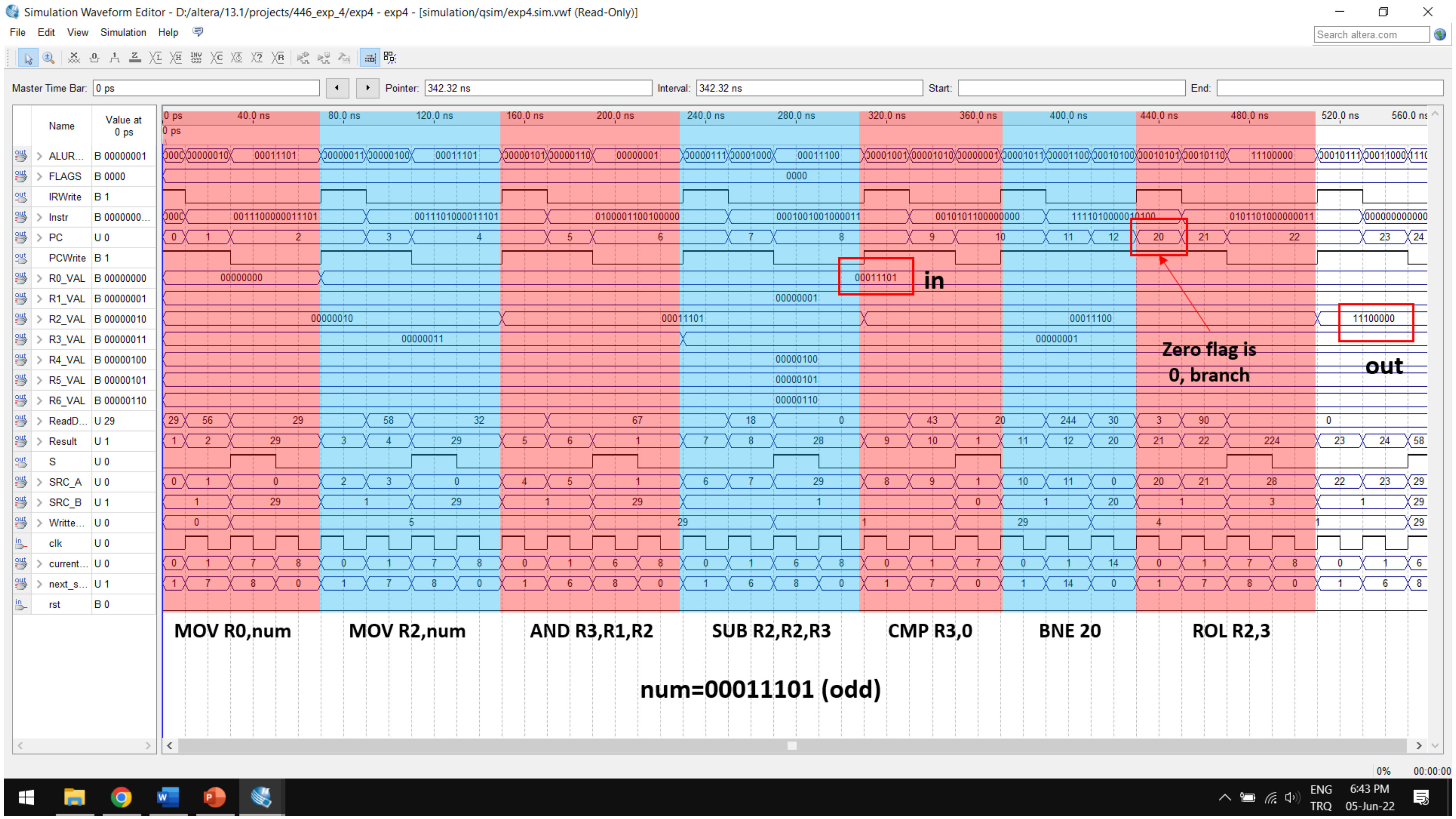
Sum of array elements. Store R0 to M[R1+26], where R0 += 1 and R0 += 2 until R0==10.



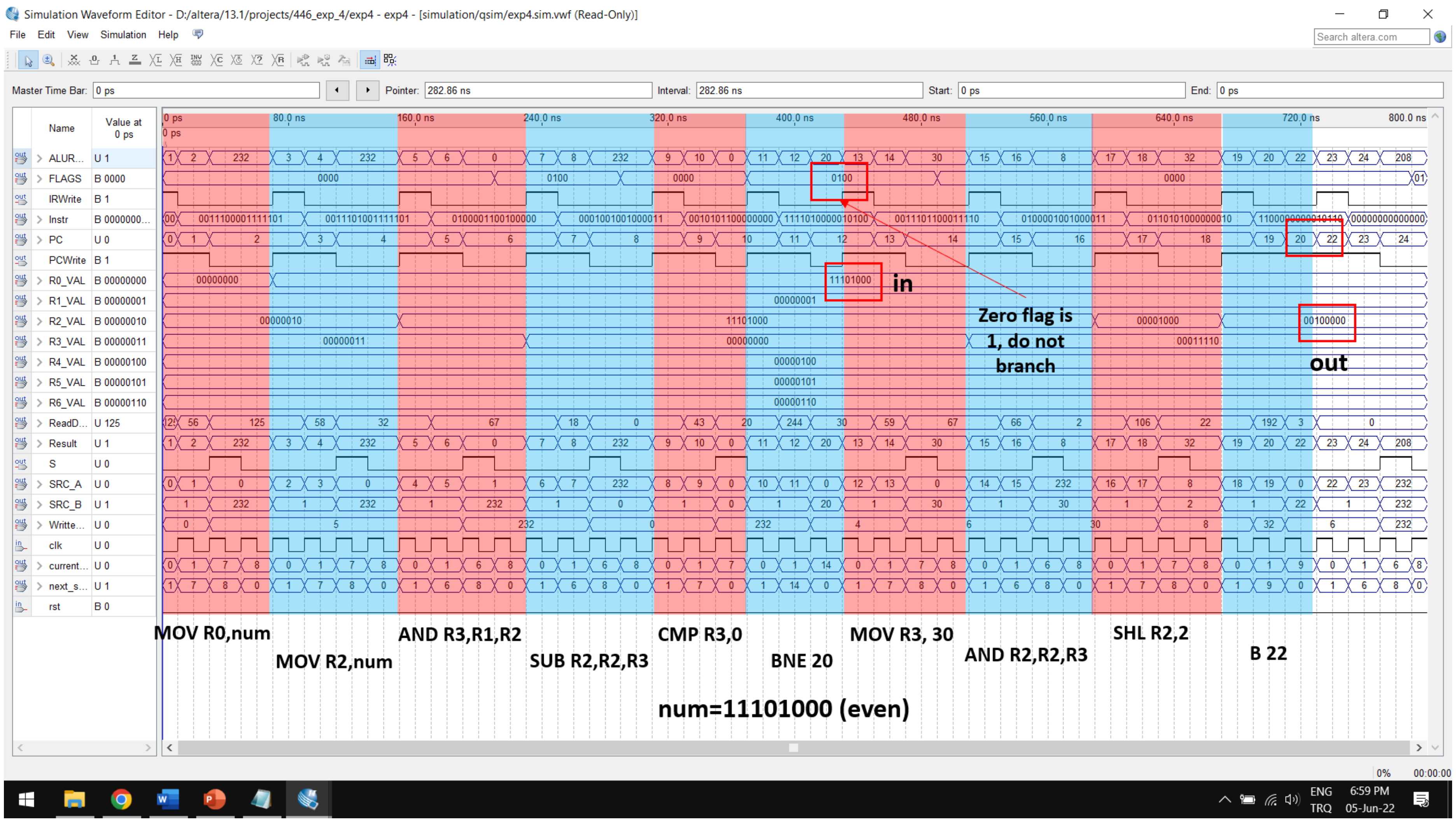
R0==10, R1==5. R1 -= 1 so that it becomes 4. Then, load R2 with M[R1+26], where R1 -= 1 and R3 += R2, until R1 == 0.



R3 calculated the sum of array elements. When R1==0, BNE 14 will not branch and program will continue.



Oddity/evenness detection.


Simulation Waveform Editor - D:/altera/13.1/projects/446_exp_4/exp4 - exp4 - [simulation/qsim/exp4.sim.vwf (Read-Only)]
in
Zero flag is 1, do not branch
out
MOV R0,num
MOV R2,num
AND R3,R1,R2
SUB R2,R2,R3
CMP R3,0
BNE 20
MOV R3, 30
AND R2,R2,R3
SHL R2,2
B 22
num=11101000 (even)

Oddity/evenness detection.