Test Codes

Test 1 (instructor's Code)

1st initializing the memory with the following values :

- M[0] = 0001
- M[1]= 0001
- M[2]= 000a
- M[60]= 430a
- M[61]= 7342

The initializing code :

Instruction	Hex Code	Expected Value
LUI R1, 0	9000	R1 = 0
ORI R1, R1, 1	2849	R1 = 1
sw $1 , 0($0)	6801	Mem[0] = 1
LUI R1, 0	9000	R1 = 0
ORI R2, R1, 1	284A	R2 = 1
ADD $1, $0, $0	0840	R1 = 0
sw $2 , 1($0)	6842	Mem[1] = 1
LUI R1, 0	9000	R1 = 0
ORI R3, R1, 10	2A8B	R3 = 10
ADD $1, $0, $0	0840	R1 = 0
sw $3 , 2($0)	6883	Mem[2] = 10
LUI R1, 1	9001	R1 = 32
ORI R3, R1, 28	2F0B	R3 = 60
ADD $1, $0, $0	0840	R1 = 0
LUI R1, 536	9218	R1 = 17152
ORI R1, R1, 10	2A89	R1 = 17162
sw $1 , 0($3)	6819	Mem[60] = 17162 = 0x430A
LUI R1, 922	939A	R1 = 29504
ORI R2, R1, 2	288A	R2 = 29506
ADD $1, $0, $0	0840	R1 = 0
sw $2 , 1($3)	685A	Mem[61] = 29506 = 0x7342
add $2 , $0 ,$0	0880	R2 = 0
add $3 , $0 ,$0	08C0	R3 = 0

The main code :

Instruction	Hex Code	Expected Value
Lui 900	9384	R1 = 28800
Addi R5, R1,13	3B4D	R5 = 28813
Xor R3, R1, R5	04CD	R3 = 13
Lw R1, 0(R0)	6001	R1 = 1
Lw R2, 1(R0)	6042	R2 = 1
Lw R3, 2(R0)	6083	R3 = 10
Addi R4, R4, 10	3AA4	R4 = 10
Sub R4, R4, R4	0B24	R4 = 0
Add R4, R2, R4	0914	R4 = 55 (Last value)
Slt R6, R2, R3	0D93	R6 = 0 (Last value)
Beq R6, R0, L1	70F0	PC = 36 (Last value)
Add R2, R1, R2	088A	R2 = 10 (Last value)
Beq R0, R0, L2	7700	PC = 31
Sw R4, 0(R0)	6804	Mem[0] = 55
Jal func	F804	PC = 41 , R7 = 38
Sll R3, R2, 6	4193	R3 = -15744 = 0XC280
ROR R6, R3, 3	58DE	R6 = 6224 = 0x1850
beq r0,r0,0	7000	PC = 40 (always)
or R5, R2, R3	0353	R5 = 10
Lw R1, 0(R0)	6001	R1 = 55
Lw R2, 5(R1)	614A	R2 = 17162 = 0x430A
Lw R3 ,6(R1)	618B	R3 = 29506 = 0x7342
And R4, R2, R3	0113	R4 = 17154 = 0x4302
Sw R4, 0(R0)	6804	Mem[0] = 17154 = 0x4302
Jr R7	1038	PC = 38

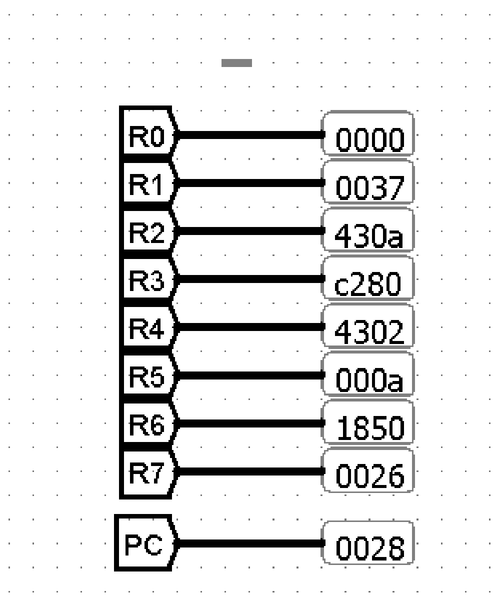
An error faced :

For the addressing mode in branch instructions the next PC should be
PC = PC + sign-extend (Imm5)

So in order to stay at the same instruction the offset should be zero so that the next PC will be the same as the current PC → PC = PC + 0

So we had to change the instruction (beq r0,r0,-1) to (beq r0,r0,0) to keep locking back to the same instruction and the program would be over.

Actual values :



Final expected values :

Register	Decimal Value	Hex Value
PC	40	0028
R0	0	0000
R1	55	0037
R2	17162	430A
R3	-15744	C280
R4	17154	4302
R5	10	000A
R6	6224	1850
R7	38	0026

Test 2 (Procedure code)

Code:

```
li $6 ,200              # Stack pointer
addi $1 ,$0 ,0          # Base address of array
addi $2 $0 10           # array size 10
JAL init_array                    # jump and link to intialize function
                        with   parameters base address and num of
                        elements
JAL sum_array           # jump and link to sum function
JAL exit                  # end program
exit:
JR $7                    # loop to itfself

#init funtion
init_array:
addi $3,$0,0             # counter=0
addi $6,$6,-1           # sp-1
sw $1,0($6)              # store base address in stack
addi $6,$6,-1
sw $2,0($6)
loop_init:
BGE $3,$2,end_init       # if counter>= sizea  end intialization
addi $4 , $0,3          #inializing value 3
sw $4,0($1)
addi $1,$1,1             #increment base address
addi $3,$3,1              #increment counter
J loop_init
end_init:
lw $2,0($6)               #return size
```

```
lw $1,1($6)            # return base address
addi $6,$6,2           # return stack pointer
JR $7                  #jump to Sum function
#sum function
sum_array:
addi $3,$0,1           #counter =1
lw $4,0($1)            # load array elements to register $4
loop_sum:
bge $3,$2,end_sum      #if counter>= size  end sum

addi $1,$1,1           # increment base address
addi $3,$3,1           #increment counter
lw $5,0($1)            # load next element to $5
add $4,$4,$5
J loop_sum
end_sum:
JR $7
```

Encoded representation:

Original Instruction	Encoded Instruction	Hex Value
LUI 6	1001000000000110	9006
ORI R6, R1, 8	0010101000001110	2A0E
ADD $1, $0, $0	0000100001000000	0840
addi $1 ,$0 ,0	0011100000000001	3801
addi $2 $0 10	0011101010000010	3A82
JAL init_array	1111100000000100	F804
JAL sum_array	1111100000010010	F812
JAL exit	1111100000000001	F801
JR $7	0001000000111000	1038
addi $3,$0,0	0011100000000011	3803
addi $6,$6,-1	0011111111110110	3FF6
sw $1,0($6)	0110100000110001	6831
addi $6,$6,-1	0011111111110110	3FF6
sw $2,0($6)	0110100000110010	6832
BGE $3,$2,end_init	1000100110011010	899A
addi $4 , $0,3	0011100011000100	38C4
sw $4,0($1)	0110100000001100	680C
addi $1,$1,1	0011100001001001	3849
addi $3,$3,1	0011100001011011	385B
J loop_init	1111011111111011	F7FB
lw $2,0($6)	0110000000110010	6032
lw $1,1($6)	0110000001110001	6071
addi $6,$6,2	0011100010110110	38B6
JR $7	0001000000111000	1038
addi $3,$0,1	0011100001000011	3843
lw $4,0($1)	0110000000001100	600C
bge $3,$2,end_sum	1000100110011010	899A
addi $1,$1,1	0011100001001001	3849
addi $3,$3,1	0011100001011011	385B
lw $5,0($1)	0110000000001101	600D
add $4,$4,$5	0000100100100101	0925
J loop_sum	1111011111111011	F7FB
JR $7	0001000000111000	1038

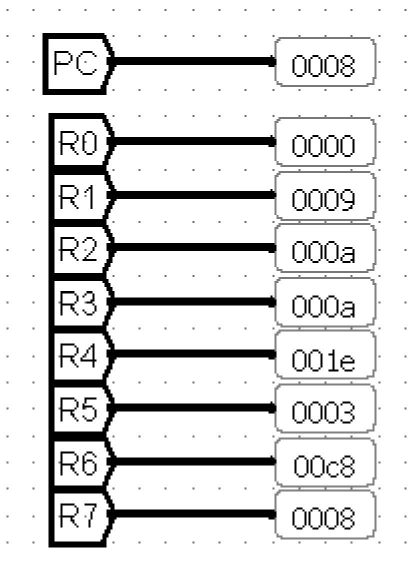
Expected Output:

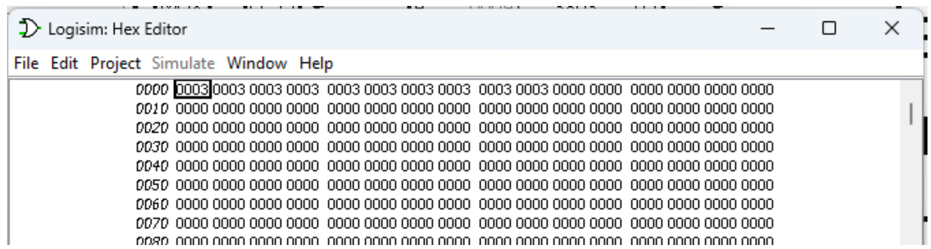
Memory | Labels

Address	Decimal Value	Hex Value
0000	3	0003
0001	3	0003
0002	3	0003
0003	3	0003
0004	3	0003
0005	3	0003
0006	3	0003
0007	3	0003
0008	3	0003
0009	3	0003
00C6	10	000A
00C7	0	0000

Register	Decimal Value	Hex Value
PC	8	0008
R0	0	0000
R1	9	0009
R2	10	000A
R3	10	000A
R4	30	001E
R5	3	0003
R6	200	00C8
R7	8	0008

Actual Output:





Test 3 (Extra code)

Code

```
   LUI   16
   ANDI $2, $1, 31
   ORI  $3, $1, 31
   XORI $4, $1, 15
   # Arithmetic operations
   ADD  $5, $2, $3
   SUB  $6, $3, $2
   # Set Less Than operations
   SLT  $7, $5, $6
   SLTU $1, $6, $5
   # Shift operations
   SLL  $2, $3, 3
   SRL  $3, $3, 2
   SRA  $4, $4, 2
   ROR  $5, $4, 1
   # Memory operations using immediate addressing
   SW   $5, 10($0)
   LW   $6, 10($0)
   # Branching
   BNE  $5, $6, diff
   BEQ  $5, $6, same
   BLT  $2, $3, less
   BGE  $3, $2, greater
diff:
   ADDI $5, $5, -5
same:
   ADDI $6, $6, 5
less:
   SUB  $2, $3, $2
greater:
   OR   $3, $3, $2
   # Jumpin
 JAL  exit
exit:
   JR   $7            # Jump to address in $7 to terminate
```

Encoded instructions

Original Instruction	Encoded Instruction	Hex Value
LUI 16	1001000000010000	9010
ANDI $2, $1, 31	0010011111001010	27CA
ORI $3, $1, 31	0010111111001011	2FCB
XORI $4, $1, 15	0011001111001100	33CC
ADD $5, $2, $3	0000100101010011	0953
SUB $6, $3, $2	0000101110011010	0B9A
SLT $7, $5, $6	0000110111101110	0DEE
SLTU $1, $6, $5	0000111001110101	0E75
SLL $2, $3, 3	0100000011011010	40DA
SRL $3, $3, 2	0100100010011011	489B
SRA $4, $4, 2	0101000010100100	50A4
ROR $5, $4, 1	0101100001100101	5865
SW $5, 10($0)	0110101010000101	6A85
LW $6, 10($0)	0110001010000110	6286
BNE $5, $6, diff	0111100100101110	792E
BEQ $5, $6, same	0111000100101110	712E
BLT $2, $3, less	1000000100010011	8113
BGE $3, $2, greater	1000100100011010	891A
ADDI $5, $5, -5	0011111011101101	3EED
ADDI $6, $6, 5	0011100101110110	3976
SUB $2, $3, $2	0000101010011010	0A9A
OR $3, $3, $2	0000001011011010	02DA
JAL exit	1111100000000001	F801
JR $7	0001000000111000	1038

Expected Output:

Memory | Labels

Address	Decimal Value	Hex Value
000A	-32703	8041

Register	Decimal Value	Hex Value
PC	23	0017
R0	0	0000
R1	0	0000
R2	-4209	EF8F
R3	-4209	EF8F
R4	131	0083
R5	-32703	8041
R6	-32698	8046
R7	23	0017

Actual Output: