ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION
PROJECT 2 REPORT

CRN : 21335

LECTURER : Deniz Turgay Altılar

GROUP MEMBERS:

150220022 : ALI EREN CIFTCI

150220033 : ISMAIL CIFTCI

SPRING 2024

Contents

1 INTRODUCTION

In this project a hardwired control unit was implemented and connected to the ALU System from the previous project. 33 operations were designed according to the given descriptions and then their control signals were implemented.

2 MATERIALS AND METHODS

The CPU system is comprised of an ALU system module from the previous project, and a hardwired control unit module. The instructions are stored in memory in little-endian order. Instructions are 16-bit long have two formats: with and without an address reference. 33 operations are described.

OPCODE (6-bit)	RSEL (2-bit)	ADDRESS (8-bit)

Table 1: Instructions with an address reference.

OPCODE (6-bit)	S (1-bit)	DSTREG(3-bit)	SREG1 (3-bit)	SREG2 (3-bit)

Table 2: Instructions without an address reference.

The RSEL is a 2-bit field that specifies which Rx from RF is used. The S is a 1-bit field that specifies whether the flags will change or not The DSTREG is a 3-bit field that specifies the destination register. The SREG1 is a 3-bit field that specifies the first source register. The SREG2 is a 3-bit field that specifies the second source register.

RSEL	REGISTER
00	R1
01	R2
10	R3
11	R4

Table 3: RSEL table.

DSTREG/SREG1/SREG2	REGISTER
000	PC
001	PC
010	SP
011	AR
100	R1
101	R2
110	R3
111	R4

Table 4: DSTREG/SREG1/SREG2 selection table.

OPCODE 0x00, BRA: PC ← PC + VALUE
OPCODE 0x01, BNE: IF Z=0 THEN PC ← PC + VALUE
OPCODE 0x02, BEQ: IF Z=1 THEN PC ← PC + VALUE
OPCODE 0x03, POP: SP ← SP + 1, Rx ← M[SP]
OPCODE 0x04, PSH: M[SP] ← Rx, SP ← SP – 1
OPCODE 0x05, INC: DSTREG ← SREG1 + 1
OPCODE 0x06, DEC: DSTREG ← SREG1 – 1
OPCODE 0x07, LSL: DSTREG ← LSL SREG1
OPCODE 0x08, LSR: DSTREG ← LSR SREG1
OPCODE 0x09, ASR: DSTREG ← ASR SREG1
OPCODE 0x0A, CSL: DSTREG ← CSL SREG1
OPCODE 0x0B, CSR: DSTREG ← CSR SREG1
OPCODE 0x0C, AND: DSTREG ← SREG1 AND SREG2
OPCODE 0x0D, ORR: DSTREG ← SREG1 OR SREG2
OPCODE 0x0E, NOT: DSTREG ← NOT SREG1
OPCODE 0x0F, XOR: DSTREG ← SREG1 XOR SREG2
OPCODE 0x10, NAND: DSTREG ← SREG1 NAND SREG2
OPCODE 0x11, MOVH: Rx[15:8] ← IMMEDIATE (8-bit)
OPCODE 0x12, LDR(16-bit): Rx ← M[AR] (AR is 16-bit register)
OPCODE 0x13, STR(16-bit): M[AR] ← Rx (AR is 16-bit register)
OPCODE 0x14, MOVL: Rx[7:0] ← IMMEDIATE (8-bit)
OPCODE 0x15, ADD: DSTREG ← SREG1 + SREG2
OPCODE 0x16, ADC: DSTREG ← SREG1 + SREG2 + CARRY
OPCODE 0x17, SUB: DSTREG ← SREG1 - SREG2

OPCODE 0x18, MOVS: DSTREG ← SREG1, Flags will change

OPCODE 0x19, ADDS: DSTREG ← SREG1 + SREG2, Flags will change

OPCODE 0x1A, SUBS: DSTREG ← SREG1 - SREG2, Flags will change

OPCODE 0x1B, ANDS: DSTREG ← SREG1 AND SREG2, Flags will change

OPCODE 0x1C, ORRS: DSTREG ← SREG1 OR SREG2, Flags will change

OPCODE 0x1D, XORS: DSTREG ← SREG1 XOR SREG2, Flags will change

OPCODE 0x1E, BX: M[SP] ← PC, PC ← Rx

OPCODE 0x1F, BL: PC ← M[SP]

OPCODE 0x20, LDRIM: Rx ← VALUE (VALUE defined in ADDRESS bits)

OPCODE 0x21, STRIM: M[AR+OFFSET] ← Rx (AR is 16-bit register) (OFFSET defined in ADDRESS bits)

2.1 CPU System

It has two inputs named Clock and Reset. Clock supplies the clock signal and Reset is used to give reset signal to the system. It has one 8-bit output named T that shows the sequence count the system is in. It contains an ArithmeticLogicUnitSystem module and a ControlUnit module. Outputs of the ControlUnit module is connected to the inputs of the ArithmeticLogicUnitSystem module.

ControlUnit: It has two inputs named Clock, Reset and a 16-bit input named IROut pulled from the ArithmeticLogicUnitSystem. It has outputs for the control signal inputs needed by ArithmeticLogicUnitSystem, as given in the following table.

CU Outputs	ALUS Inputs
RF_OutASel	RF_OutASel
RF_OutBSel	RF_OutBSel
RF_FunSel	RF_FunSel
RF_RegSel	RF_RegSel
ALU_WF	ALU_WF
ALU_FunSel	ALU_FunSel
ARF_OutCSel	ARF_OutCSel
ARF_OutDSel	ARF_OutDSel
ARF_FunSel	ARF_FunSel
ARF_RegSel	ARF_RegSel
IR_LH	IR_LH
IR_Write	IR_Write
Mem_WR	Mem_WR
Mem_CS	Mem_CS
MuxCSel	MuxCSel
MuxASel	MuxASel
MuxBSel	MuxBSel

Table 5: ControlUnit (CU) outputs connected to ArithmeticLogicUnitSystem (ALUS) inputs.

2.2 RTL Description

33 operations were designed in RTL according to the given descriptions and according to these designs their control signals were implemented. Using a decoder the opcodes were transformed into a 33 digit binary code named D. Using D and the decoded version of the sequence counter named T as control signals we can describe the operations in RTL.

2.2.1 ALL D

T[0]: IROut(0-7)<-M[PC], PC<-PC+1

T[1]: IROut(8-15)<-M[PC], PC<-PC+1

T[2]: D<-Decode[IROut(10-15], S<-IROut(9)

2.2.2 D[0]|D[1]|D[2]:

T[2]&D[1]&Z: SC<-0

T[2]&D[2]&!Z: SC<-0

T[2]: S1<-IROut(0-7)

T[3]: S2<-PC

T[4]: PC<-S1+S2, S1<-0, S2<-0, SC<-0

D[3]:

T[2]: Rx(0-7)<-M[SP], SP<-SP+1

T[3]: Rx(8-15)<-M[SP], SP<-SP+1, SC<-0

D[4]:

T[2]: SP<-SP-1

T[3]: M[SP]<-Rx(0-7), SP<-SP-1

T[4]: M[SP]<-Rx(8-15), SC<-0

2.2.3 D[5]|D[6]:

T[2]: DSTREG<-SREG1

T[3]&D[5]]: DSTREG<-DSTREG+1, SC<-0

T[3]&D[6]]: DSTREG<-DSTREG-1, SC<-0

2.2.4 D[7]|D[8]|D[9]|D[10]|D[11]|D[14]|D[24]:

for D[7] OP: LSL

for D[8] OP: LSR

for D[9] OP: ASR

for D[10] OP: CSL

for D[11] OP: CSR

for D[14] OP: NOT

for D[24] OP: ID, ALU_WF<-S

SREG1 in RF

T[2]: DSTREG<- OP SREG1, SC<-0

SREG1 in ARF, DSTREG in ARF

T[2]: S1<-SREG1

T[3]: DSTREG<- OP S1, S1<-0, SC<-0

SREG1 in ARF, DSTREG in RF

T[2]: S1<-SREG1

T[3]: DSTREG<- OP S1

T[3]: S1<-0, SC<-0

2.2.5 D[12]|D[13]|D[15]|D[16]|D[21]|D[22]|D[23]|D[25]|D[26]|D[27]|D[28]|D[29]:

for D[12] OP: AND

for D[13] OP: OR

for D[15] OP: XOR

for D[16] OP: NAND

for D[21] OP: ADD

for D[22] OP: ADD with carry

for D[23] OP: SUB

for D[25] OP: ADD, ALU_WF<-S

for D[26] OP: SUB, ALU_WF<-S

for D[27] OP: AND, ALU_WF<-S

for D[28] OP: OR, ALU_WF<-S

for D[29] OP: XOR, ALU_WF<-S

SREG1 in ARF, SREG2 in ARF, DSTREG in ARF

T[2]: S1<-SREG1

T[3]: S2<-SREG2

T[4]: DSTREG<- S1 OP S2, S1<-0, S2<-0, SC<-0

SREG1 in ARF, SREG2 in ARF, DSTREG in RF

T[2]: S1<-SREG1

T[3]: S2<-SREG2

T[4]: DSTREG<- S1 OP S2

T[5]: S1<-0, S2<-0, SC<-0

SREG1 in ARF, SREG2 in RF, DSTREG in ARF

T[2]: S1<-SREG1

T[3]: DSTREG<-S1 OP SREG2, S1<-0, SC<-0

SREG1 in ARF, SREG2 in RF, DSTREG in RF

T[2]: S1<-SREG1

T[3]: DSTREG<-S1 OP SREG2

T[4]: S1<-0, SC<-0

SREG1 in RF, SREG2 in ARF, DSTREG in ARF

T[2]: S2<-SREG2

T[3]: DSTREG<-SREG1 OP S2, S2<-0, SC<-0

SREG1 in RF, SREG2 in ARF, DSTREG in RF

T[2]: S2<-SREG2

T[3]: DSTREG<-SREG1 OP S2

T[4]: S2<-0, SC<-0

SREG1 in RF, SREG2 in RF, DSTREG in ARF/RF

T[2]: DSTREG<-SREG1 OP SREG2, SC<-0

2.2.6 D[17]|D[20:

T[2]&D[17]]: DSTREG(7-0)<-IR(0-7), SC<-0

T[2]&D[20]]: DSTREG(15-8)<-IR(0-7), SC<-0

2.2.7 D[18]:

T[2]: Rx(0-7)<-M[AR], AR<-AR+1

T[3]: Rx(8-15)<-M[AR], SC<-0

2.2.8 D[19]:

T[2]: M[AR]<-Rx(0-7), AR<-AR+1

T[3]: M[AR]<-Rx(8-15) SC<-0

2.2.9 D[30]:

T[2]: S1<-PC

T[3]: M[SP]<-S1(0-7), SP<-SP-1

T[4]: M[SP]<-S1(8-15)

T[5]: PC<-Rx

2.2.10 D[31]:

T[2]: SP<-SP+1

T[3]: PC(0-7)<-M[SP], SP<-SP+1

T[4]: PC(8-15)<-M[SP], SC<-0

2.2.11 D[32]:

T[2]: Rx<-IR(0-7), SC<-0

2.2.12 D[33]:

T[2]: S1<-IR(0-7)

T[3]: S2<-AR

T[4]: AR<-S1+S2, S1<-0, S2<-0

T[5]: M[AR]<-Rx(0-7), AR<-AR+1

T[6]: M[AR]<-Rx(8-15), SC<-0

2.3 Task Distribution

We did not make a definite distribution of responsibilities at the beginning of the project however contributions from each member came out to be as follows: Code writing for modules, Ali Eren CIFTCI; Code refinement and fixing, Ali Eren CIFTCI and Ismail CIFTCI; Report writing, Ismail CIFTCI; Final checks, Ali Eren CIFTCI and Ismail CIFTCI.

3 RESULTS

3.1 Hardware

These were the hardware designs and some simulation excerpts that we acquired from our modules using Vivado.

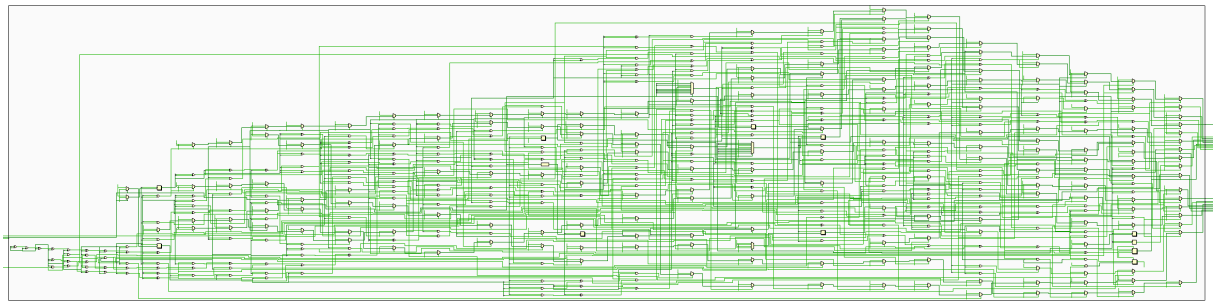

Figure 1: Control Unit

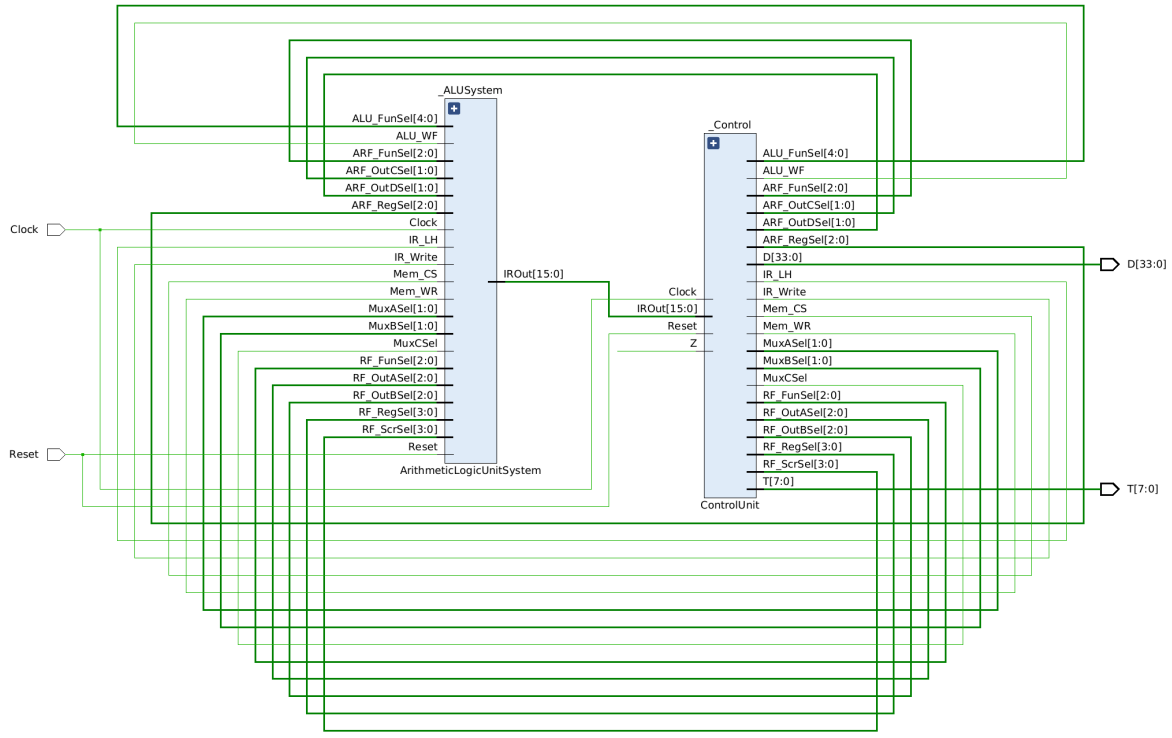

Figure 2: CPU System

3.2 Simulation

To test our code we wrote the given program in the following figure into our ram.mem file and the ran the simulation for 26300ns as this program needs this much time end.

```
1   28 //0028 BRA 0x28 (Branch to PC+40)
2   00
3   00
4   00
```

Figure 3: Implemented program beginning.

```
41   00
42   00
43   00 //4400 MOVH R1 (branched to here)
44   44
45   0A //500A MOVL R1
46   50
47   00 //4500 MOVH R2
48   45
49   00 //5100 MOVL R2
50   51
51   00 //4600 MOVH R3
52   46
53   B0 //52B0 MOVL R3
54   52
55   F0 //62F0 MOVS AR R3
56   62
57   00 //4A00 LDR R3
58   4A
59   6E //556E ADD R2 R2 R3
60   55
61   D8 //14D8 INC AR AR
62   14
63   20 //1920 DEC R1 R1
64   19
65   20 //6320 MOVS R1 R1
66   63
67   F4 //04F4 BNE 0xF4 (-12 in decimal)
68   04
69   D8 //14D8 INC AR AR
70   14
71   00 //4D00 STR R2
72   4D
73   00
74   00
```

Figure 4: Implemented program continued (row count starts from 1).

Excerpts fromt the results of some operations from the program are in the following figures.

```
Output Values:
T:   1
Address Register File: PC:     0, AR:     0, SP:   255
Instruction Register :     x
Register File Registers: R1:     0, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: x, N: x, C: x, O: x
ALU Result: ALUOut:     x
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   2
Address Register File: PC:     1, AR:     0, SP:   255
Instruction Register :     X
Register File Registers: R1:     0, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: x, N: x, C: x, O: x
ALU Result: ALUOut:     x
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   4
Address Register File: PC:     2, AR:     0, SP:   255
Instruction Register :    40
Register File Registers: R1:     0, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: x, N: x, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   8
Address Register File: PC:     2, AR:     0, SP:   255
Instruction Register :    40
Register File Registers: R1:     0, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:    40, S2:     0, S3:     0, S4:     0
ALU Flags: Z: x, N: x, C: x, O: x
ALU Result: ALUOut:    40
RAM_DATA[196] _ SONUC:   0


Output Values:
T:  16
Address Register File: PC:     2, AR:     0, SP:   255
Instruction Register :    40
Register File Registers: R1:     0, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:    40, S2:     2, S3:     0, S4:     0
ALU Flags: Z: x, N: x, C: x, O: x
ALU Result: ALUOut:    42
RAM_DATA[196] _ SONUC:   0
```

Figure 5: BRA 0x28

```
Output Values:
T:   1
Address Register File: PC:    56, AR:   176, SP:   255
Instruction Register : 25328
Register File Registers: R1:    10, R2:     0, R3:   176, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   2
Address Register File: PC:    57, AR:   176, SP:   255
Instruction Register : 25088
Register File Registers: R1:    10, R2:     0, R3:   176, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   4
Address Register File: PC:    58, AR:   176, SP:   255
Instruction Register : 18944
Register File Registers: R1:    10, R2:     0, R3:   176, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   8
Address Register File: PC:    58, AR:   177, SP:   255
Instruction Register : 18944
Register File Registers: R1:    10, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0
```

Figure 6: LDR R3

```
Output Values:
T:   1
Address Register File: PC:    58, AR:   177, SP:   255
Instruction Register : 18944
Register File Registers: R1:    10, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   2
Address Register File: PC:    59, AR:   177, SP:   255
Instruction Register : 19054
Register File Registers: R1:    10, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   4
Address Register File: PC:    60, AR:   177, SP:   255
Instruction Register : 21870
Register File Registers: R1:    10, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0
```

Figure 7: ADD R2, R2, R3

```
Output Values:
T:   1
Address Register File: PC:    60, AR:   177, SP:   255
Instruction Register : 21870
Register File Registers: R1:    10, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   2
Address Register File: PC:    61, AR:   177, SP:   255
Instruction Register : 21976
Register File Registers: R1:    10, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   4
Address Register File: PC:    62, AR:   177, SP:   255
Instruction Register :  5336
Register File Registers: R1:    10, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   8
Address Register File: PC:    62, AR:   177, SP:   255
Instruction Register :  5336
Register File Registers: R1:    10, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0
```

Figure 8: INC AR, AR (first)

```
Output Values:
T:   1
Address Register File: PC:    66, AR:   178, SP:   255
Instruction Register : 25376
Register File Registers: R1:     9, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     9, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     9
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   2
Address Register File: PC:    67, AR:   178, SP:   255
Instruction Register : 25588
Register File Registers: R1:     9, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     9, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     9
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   4
Address Register File: PC:    68, AR:   178, SP:   255
Instruction Register :  1268
Register File Registers: R1:     9, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1:     9, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:     9
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   8
Address Register File: PC:    68, AR:   178, SP:   255
Instruction Register :  1268
Register File Registers: R1:     9, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1: 65524, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut: 65524
RAM_DATA[196] _ SONUC:   0


Output Values:
T:  16
Address Register File: PC:    68, AR:   178, SP:   255
Instruction Register :  1268
Register File Registers: R1:     9, R2:     0, R3:     0, R4:     0
Register File Scratch Registers: S1: 65524, S2:    68, S3:     0, S4:     0
ALU Flags: Z: 0, N: 0, C: x, O: x
ALU Result: ALUOut:    56
RAM_DATA[196] _ SONUC:   0
```

Figure 9: BNE 0xF4

```
Output Values:
T:   1
Address Register File: PC:    70, AR:   197, SP:   255
Instruction Register :  5336
Register File Registers: R1:     0, R2:    45, R3:     9, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 1, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   2
Address Register File: PC:    71, AR:   197, SP:   255
Instruction Register :  5120
Register File Registers: R1:     0, R2:    45, R3:     9, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 1, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   4
Address Register File: PC:    72, AR:   197, SP:   255
Instruction Register : 19712
Register File Registers: R1:     0, R2:    45, R3:     9, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 1, N: 0, C: x, O: x
ALU Result: ALUOut:    45
RAM_DATA[196] _ SONUC:   0


Output Values:
T:   8
Address Register File: PC:    72, AR:   196, SP:   255
Instruction Register : 19712
Register File Registers: R1:     0, R2:    45, R3:     9, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 1, N: 0, C: x, O: x
ALU Result: ALUOut:    45
RAM_DATA[196] _ SONUC:  45
```

Figure 10: STR R2

```
Output Values:
T:   1
Address Register File: PC:    72, AR:   196, SP:   255
Instruction Register : 19712
Register File Registers: R1:     0, R2:    45, R3:     9, R4:     0
Register File Scratch Registers: S1:     0, S2:     0, S3:     0, S4:     0
ALU Flags: Z: 1, N: 0, C: x, O: x
ALU Result: ALUOut:     0
RAM_DATA[196] _ SONUC:  45
```

Figure 11: END of program

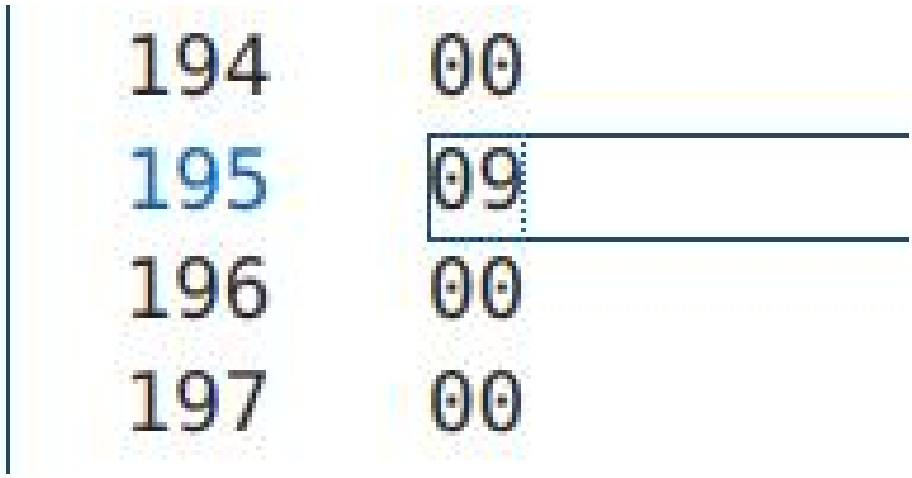

Figure 12: This program is supposed to write the sum of values 0x1 to 0x9, which were contained within the cells from 0xB0 to 0xC3 (row count starts from 1), to the cells 0xC4 and 0xC5 in little endian form.

4 DISCUSSION

Using Verilog we described the control unit of the intended CPU in functional terms, connected it to the Arithmetic Logic Unit system and procured a design in Vivado using real-world hardware components such as wires, logic gates, buffers, latches, flip-flops, and multiplexers. Our system was simulated again in Vivado and the results were compared against given output examples and our understanding of the described system written in Verilog. We considered the results of these tests in correcting the designed system.

5 CONCLUSION

In this project the foremost difficulty we faced was in collecting the conditions for the control signals. For example if we had a mistake in one of our designs for an operation we would need to change a lot of components relatred to it. And because of the vastness of the code these changes were hard to keep track of. We acquired a better understanding of control signal hardwiring and a better understanding of RTL explanations of general operations.

REFERENCES