Project 3: Non-Pipelined Control Unit

Presented by:
Devin Burnside
Aaron Sens

EECE3026
Due Date: 3/7/2017

Table of Contents

I. Specifications

The basic characteristics of this machine are: word size of 16 bits, data bus size of 16 bits, byte addressable memory, 16-bit program status word (PSW), 16 instructions (14 user and 2 privileged) 8 general purpose registers, 16-bit program counter, 16-bit countdown timer, and 2's complement number representation.

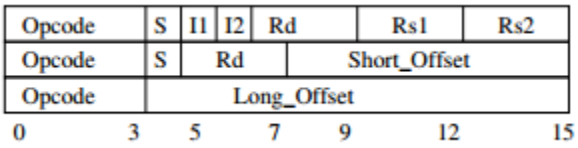

Table 1: Instruction Format

I1/I2	Operand
I1 = 0	OP1 = Reg[Rs1];
I1 = 1	OP1 = MM[Reg[Rs1]];
I2 = 0	OP2 = Reg[Rs2];
I2 = 1	OP2 = MM[Reg[Rs2]];

Table 2: Operand Location Determination Bits

Name	Opcode	Description
ADD	0	GPR[Rd] = OP1 + OP2
SUB	1	GPR[Rd] = OP1 - OP2
AND	2	GPR[Rd] = OP1 **and** OP2
SHL	3	GPR[Rd] = shift_left(OP1) by $OP2_{3-0}$
SHRA	4	GPR[Rd] = shift_right(OP1) by $OP2_{3-0}$
OR	5	GPR[Rd] = OP1 **or** OP2
NOT	6	GPR[Rd] = **not** MM[PC + Short_Offset]
LD	7	GPR[Rd] = MM[PC + Short_Offset]
ST	8	MM[PC + Short_Offset] = GPR[Rd]
BRN	9	if CC.N then PC = PC + Long_Offset
BRZ	10	if CC.Z then PC = PC + Long_Offset
BR	11	PC = PC + Long_Offset
JSR	12	GPR[Rd] = PC; PC = PC + Short_Offset
RTS	13	PC = GPR[Rd] + Short_Offset
CLK	14	Set timer to MM[PC + Long_Offset]
LPSW	15	PSW = MM[PC + Long_Offset]

Table 3: Instructions and their Semantics

II. High Level Design

The objective of this project is to design and implement a single bus non-pipelined control unit with the instruction set predetermined in the specifications.

A high level block diagram of the machine is shown in **Figure 1**.

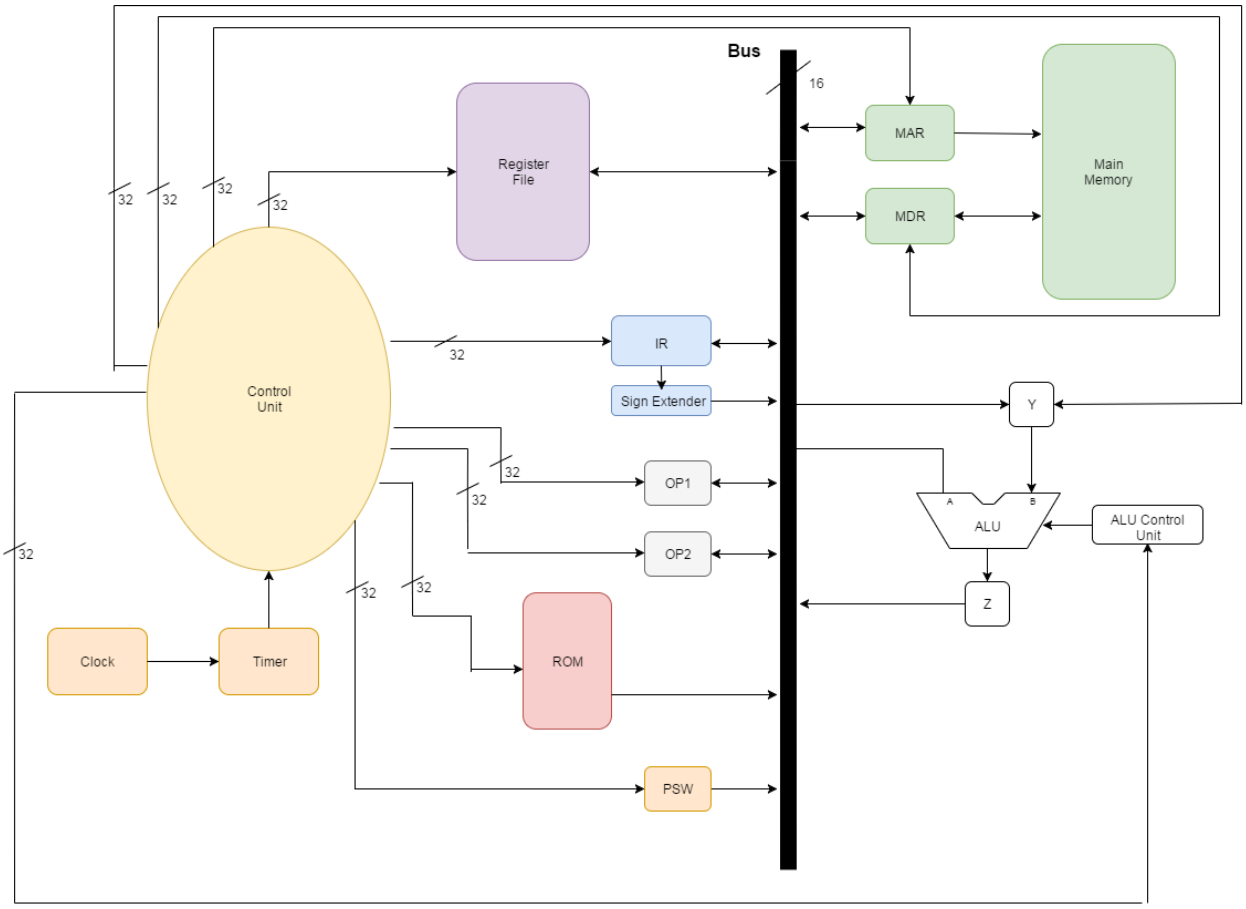

Figure 1: High Level Block Diagram of Machine

III. Component Explanation

A. Register

Several registers are used throughout the design of the machine. These registers are used to store binary data. A register is simply a group of flip-flops that are all on the same clock. Each flip-flop stores one bit. **Figure 2** below is a 16-bit register that uses 16 D flip-flops to store each bit of the data.

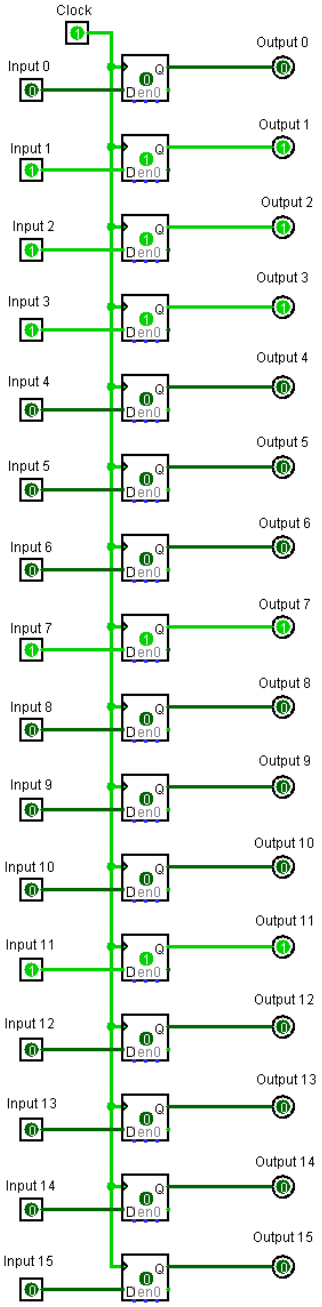

Figure 2: 16-Bit Register

B. Register File

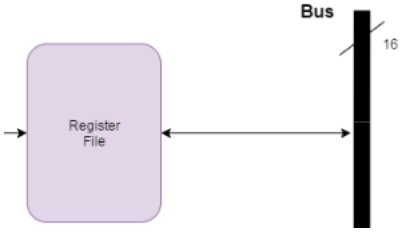

Figure 3: Register File Block Diagram

The register file is the array of general purpose registers in the machine. Data from the bus can be written into the registers or read from the registers and sent onto the bus. The register being read or written to is specified by Rd, Rs1, or Rs2 in the instructions. GPR R0 hold the constant read only value 0 and GPR R7 holds the Program Counter. An extra register Reg8 and an adder are attached to the GPR bank for PC incrementation purposes. Whenever the PC is incremented it uses the adder in the register file, and stores the value in Register 8 until the instruction is complete. The full implementation of the register file is shown below in **Figure 4**.

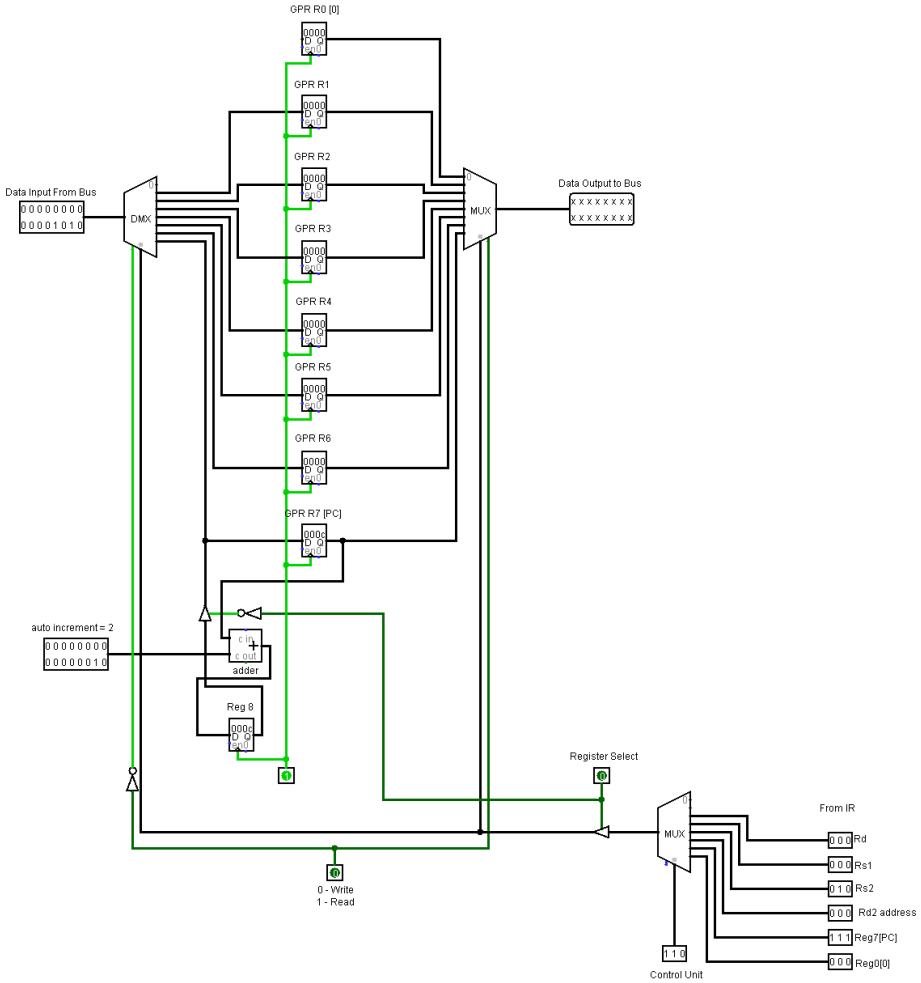

Figure 4: Internals of Register File

C. Main Memory

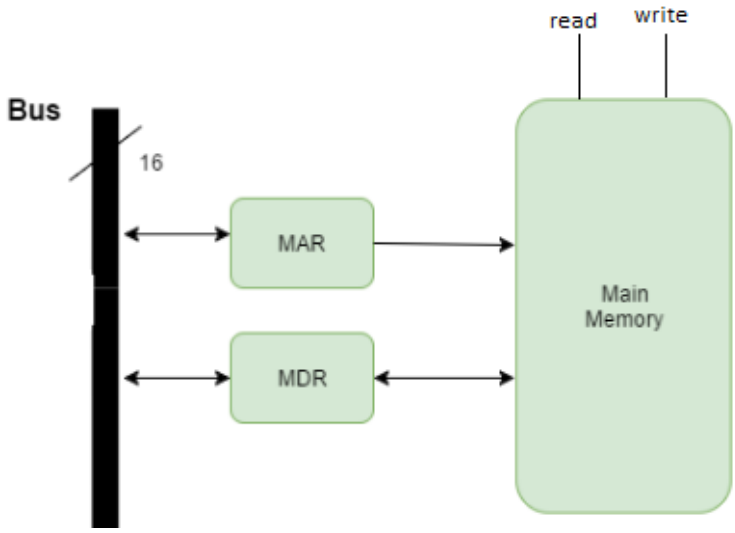

Figure 5: Main Memory Block Diagram

The 64-kilobyte main memory has four inputs and one output as seen in **Figure 5**. The MAR (Memory Address Register) inputs the memory address to read the data from. Main memory then stores the data in the MDR (Memory Data Register) which then can be sent back onto the bus. Data can also be input from the bus into the MDR to be written into the memory. The read and write inputs are controlled by the control unit.

D. Assorted Registers

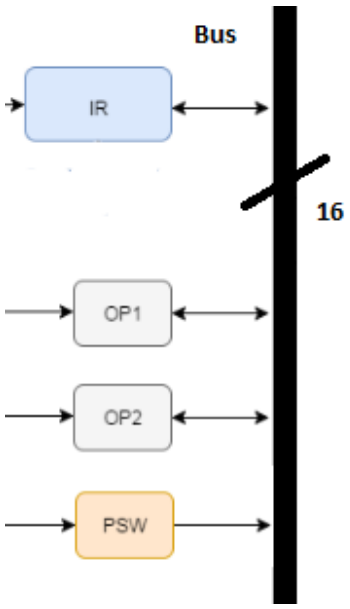

Figure 6: Assorted Registers Block Diagram

- IR (Instruction Register) - The instruction register receives and outputs the instruction data bits from main memory.

- PSW (Program Status Word) - The PSW stores 16-bits of binary data. The first two bits are condition code bits Z and N. The third bit is the P bit which denotes execution in privileged or user mode.

- OP1 (Operand 1) - Temporary register that stores and outputs operand one while executing the instruction.

- OP2 (Operand 2) - Temporary register that stores and outputs operand two while executing the instruction.

E. ROM

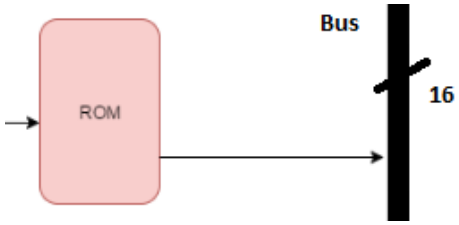

Figure 7: ROM Block Diagram

ROM (Read-Only Memory) is a memory unit that is hardwired and cannot be written to. In this machine the control unit selects the register to read and place on the bus. In this machine the ROM holds 8 constants (0,2,4,6,8,10,12,14) which are utilized during program check violation and timeout operations. The ROM operates fast enough so that memory operations do not have to delay a clock cycle like main memory does.

F. Counter

Counters are implemented using and gates and JK flip flops. During each clock cycle the counter increases if the input value is 1. In this machine 2 bit and 4 bit counters are used to iterate through control unit substates.

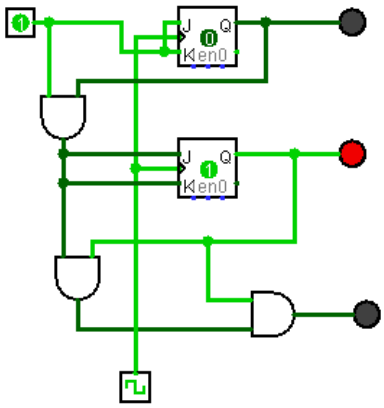

Figure 8: 2-Bit Counter

G. Sign Extender

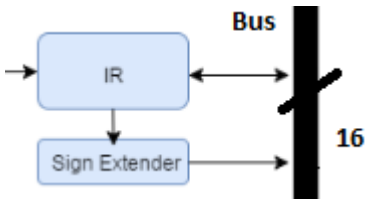

Figure 9: Sign Extender Block Diagram

The sign extender extends the short and long offsets from the instruction register to 16-bits. In the IR, the short_offset starts at bit 8 and ends at bit 15, therefore the offset needs 8 more bits. To add these bits, bit 8 is set to bit 0, then continues through bit 15 now set to bit 7. Since bit 7 is now the leftmost bit, it is the sign bit, therefore this bit and the 8 more bits needed are sign extended through bit 15 by setting them to the same value. The same process applies to the long_offset except here bit 4 is set to bit 0 and bit 11 is the sign bit where the sign extension occurs. This sign extender is implemented in **Figure 10**.

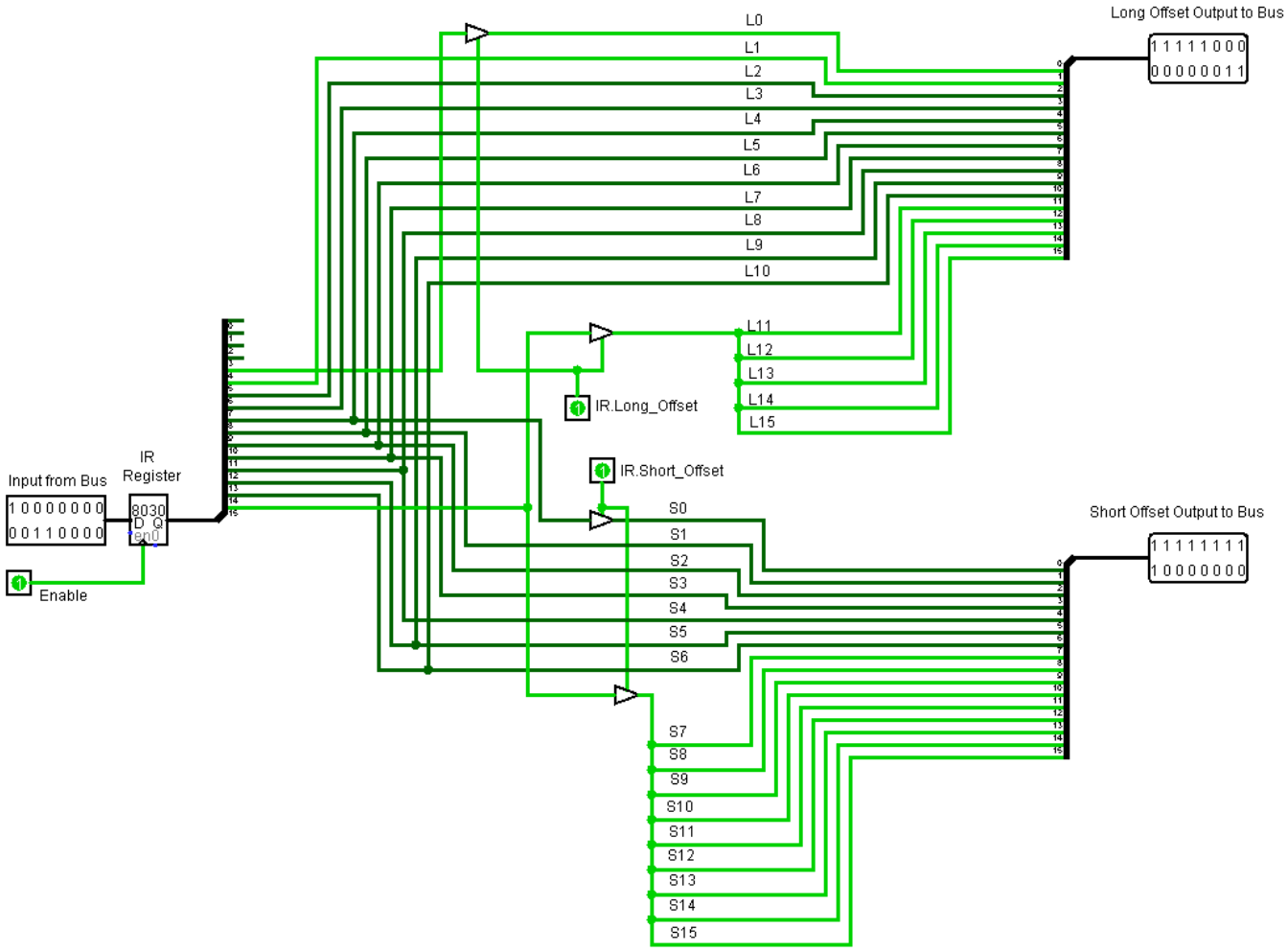

Figure 10: Sign Extender

H. ALU

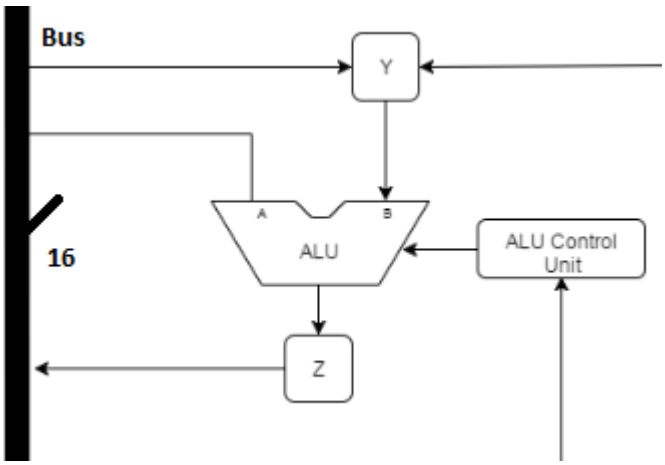

Figure 11: ALU Block Diagram

The ALU in **Figure 11** performs the arithmetic and logical operations in the system. In this design, it can perform one of seven operations as shown in **Table 4.** Each operation is completed in parallel and then one of these operation's output is selected using a multiplexer and sent to the Z register as shown in **Figure 12**.

Selector			**Operation**
0	0	0	Do Nothing
0	0	1	NOT
0	1	0	AND
0	1	1	OR
1	0	0	Add
1	0	1	Subtract
1	1	0	Logical Shift Left
1	1	1	Arithmetic Shift Right

Table 4: Operations and Selector Bits of ALU
(controlled by ALU control unit- refer to Figure 20)

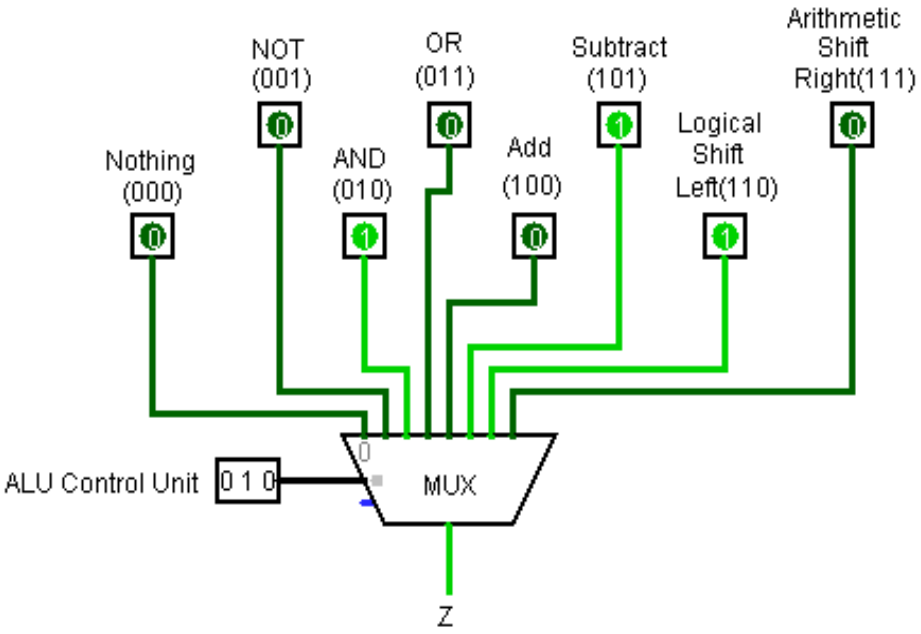

Figure 12: Internals of ALU Diagram

The NOT operation in the ALU inverts each A input bit and then outputs the result. This means for each input that is one will be output as zero and each input that is zero will have an output of one.

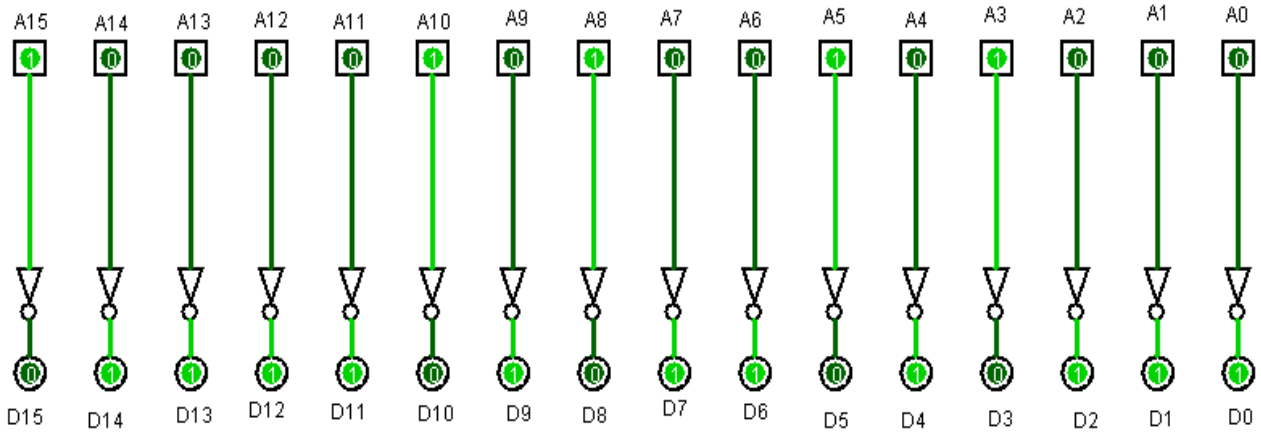

Figure 13: Logical NOT Operation in ALU

The AND operation in the ALU compares each of the 16 input bits of A and B individually and outputs the result. If both A and B input bits are one for the corresponding bit, the output is then one. Otherwise, the output is zero.

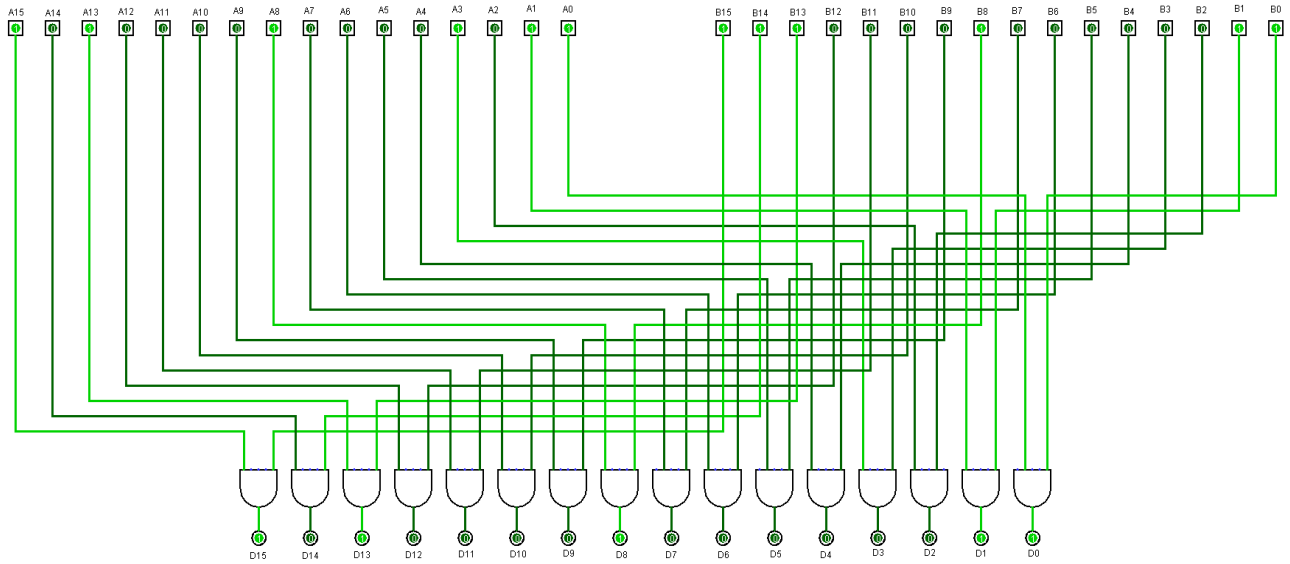

Figure 14: Logical AND Operation in ALU

The OR operation in the ALU compares each bit of the A and B inputs and then outputs the result. In an OR gate if one either A or B is one, then the output will be one. Otherwise, the output is zero.

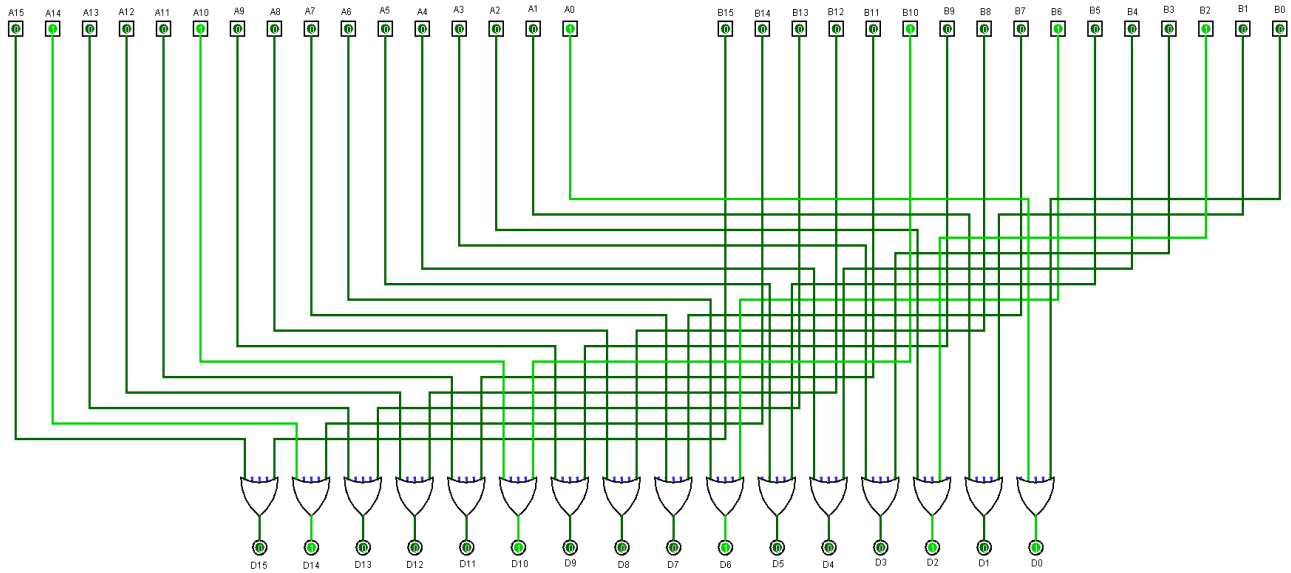

Figure 15: Logical OR Operation in ALU

The add and subtract circuit in **Figure 16** and **Figure 17** is a carry look-ahead adder that performs 2's complement addition or subtraction on the 16-bit A and B inputs. The design below shows the first four bits of both of these inputs. The circuit is able to toggle between addition and subtraction by inverting the B bits and sending them through an XOR gate with an add/subtract signal. This signal is low when adding, and high for subtracting by adding a one to the most significant bit for a 2's complement output.

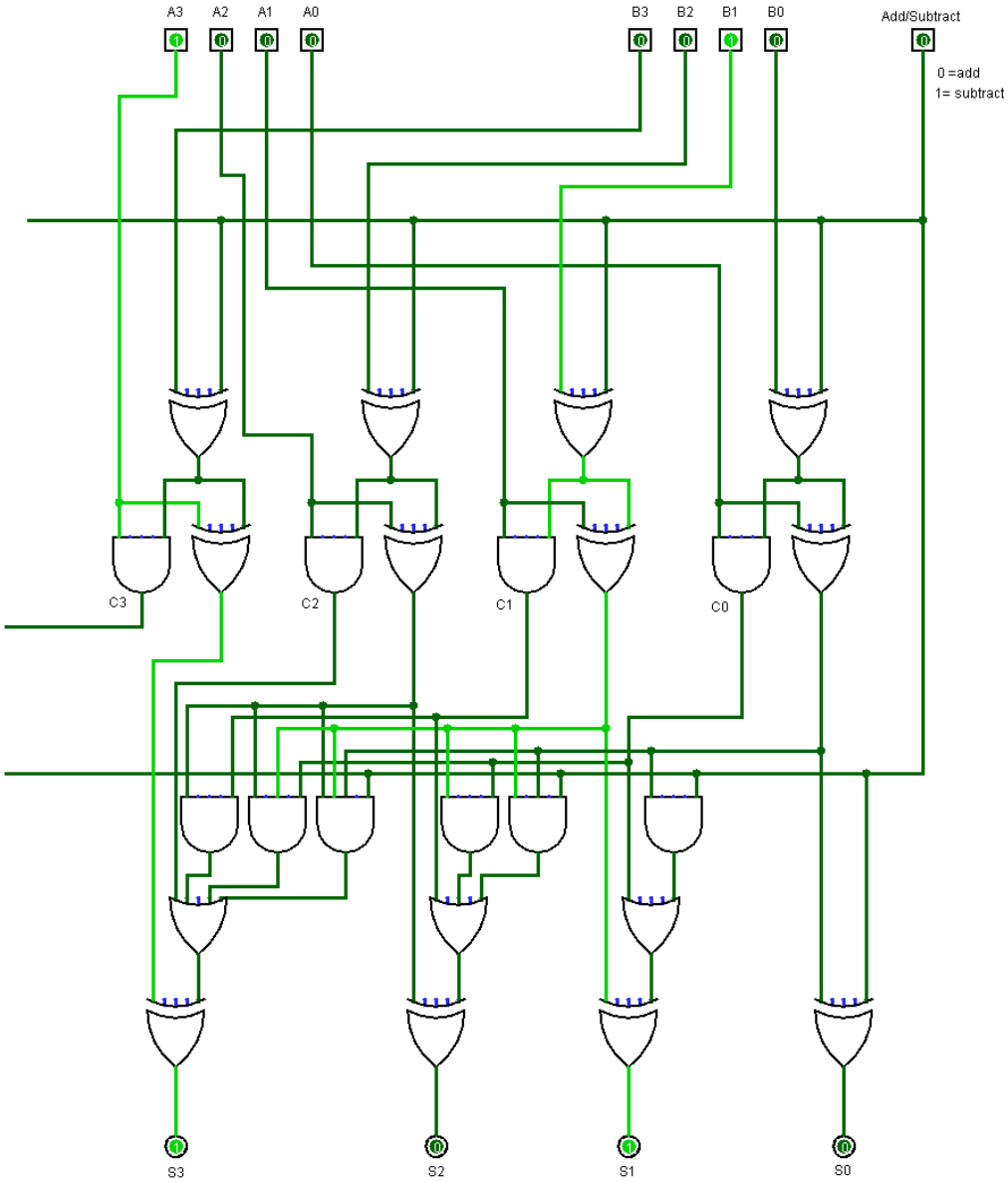

Figure 16: Add Operation in ALU

In the add operation, the A3 bit equals 8 in decimal and is being added with B2 which is 2 in decimal. The output is 1010 which is equal to ten in binary.

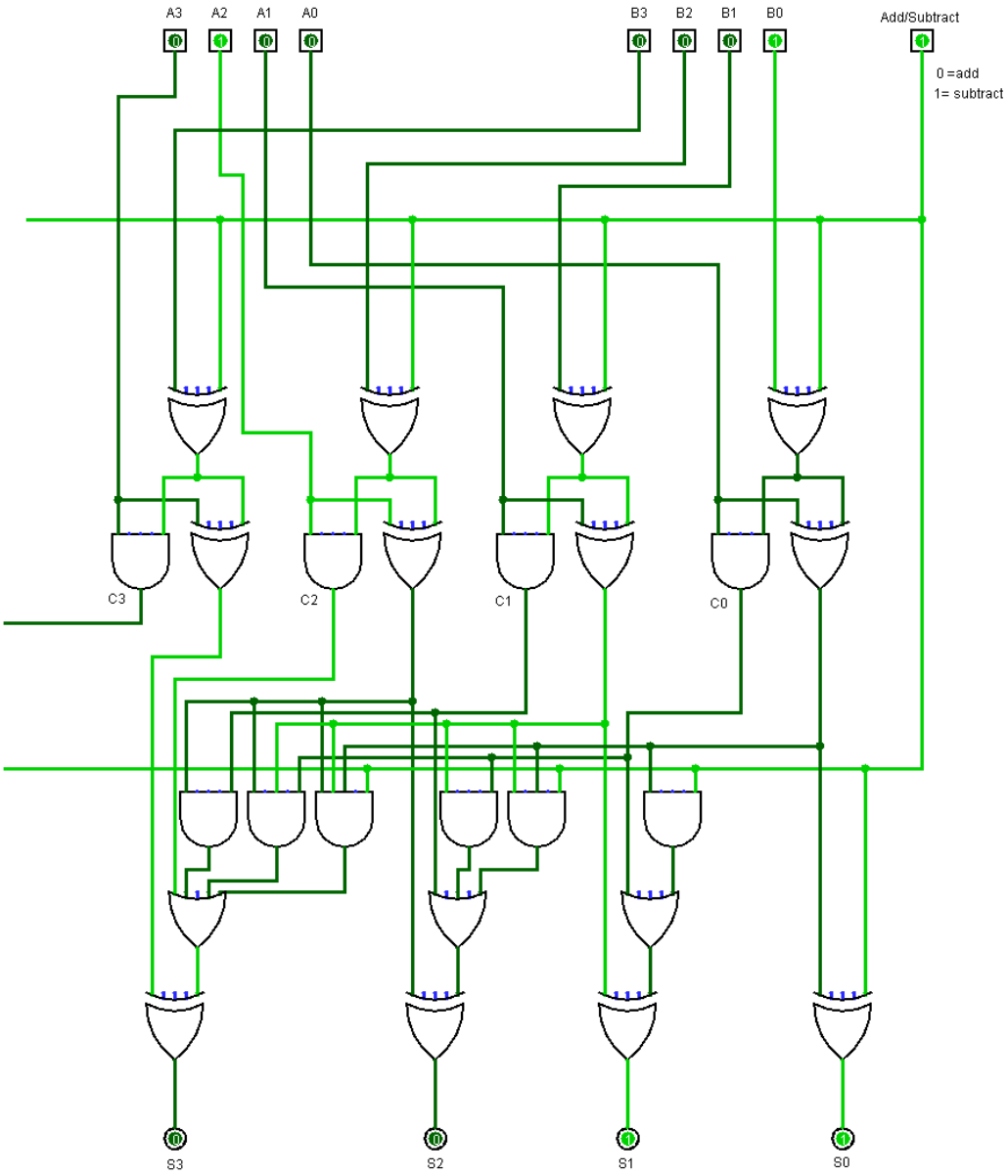

Figure 17: Subtract Operation in ALU

In the subtract operation above A2 subtracts B0. In decimal, this operation is 4-1 therefore the output is 3 (0011) in this specific simulation.

The logical shift left operation is implemented by using sixteen multiplexers. Each possible location of the individual bits is fed through the multiplexers where the four bits determine how far to shift the specific input bit of A. An enable is also placed on each multiplexer to turn the circuit on and off.

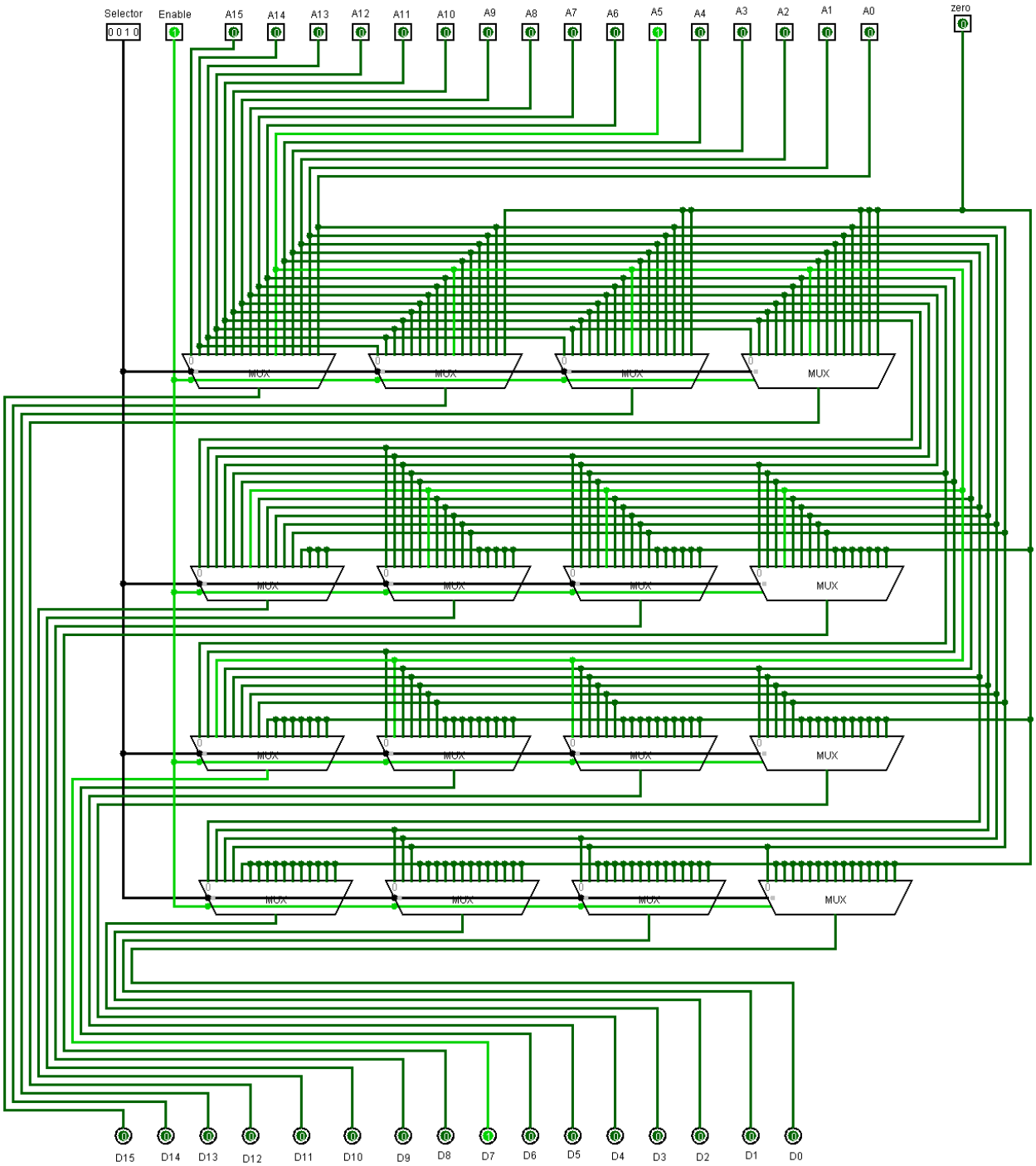

Figure 18: Logical Shift Left Operation in ALU

The arithmetic right shift used the most significant bit of A to be routed to the corresponding multiplexers. This causes the shifted bits to match the most significant bit of input A making the design arithmetic rather than logical.

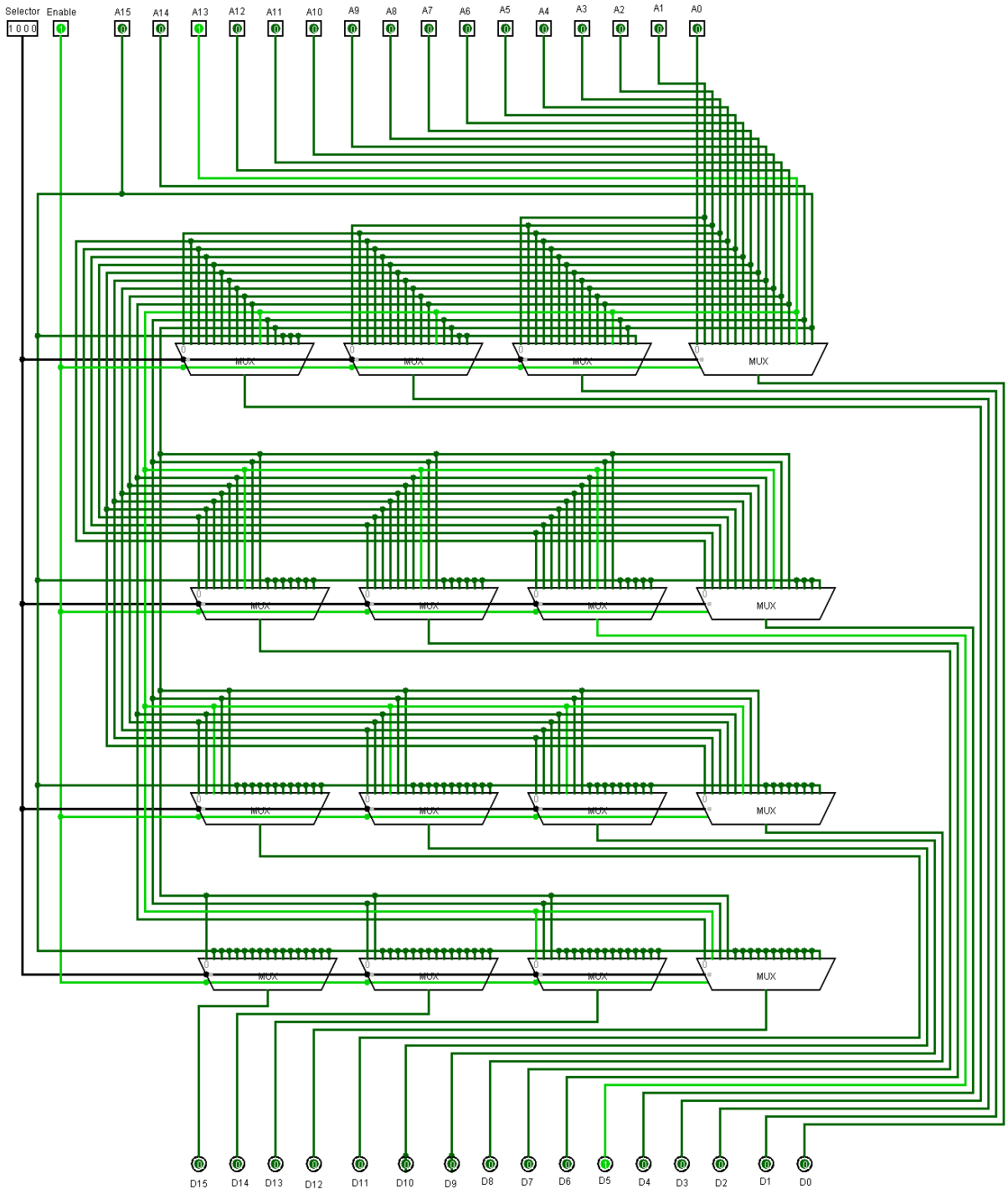

Figure 19: Arithmetic Right Shift Operation in ALU

I. ALU Control Unit

The ALU control unit specifies what operation the ALU performs. It receives an instruction from the main control unit enabling the ALU operation (ALU/op) and determines what operation to perform based on the current opcode using combinational logic.

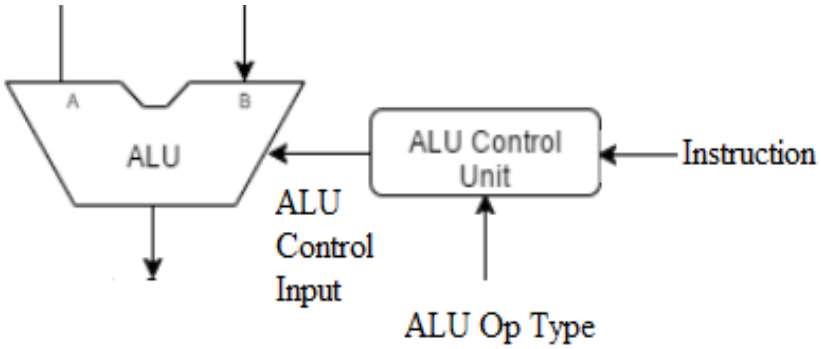

Figure 20: ALU Control Unit Block Diagram

J. Tri-State Buffer

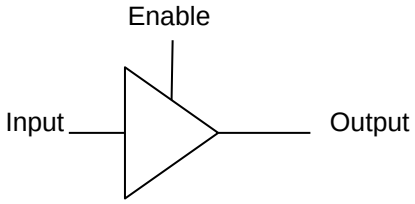

Figure 21: Tri-State Buffer

The tri state buffer is used to either input or output values between the data path and the bus, depending on control signal enables.

K. Timer

The countdown timer in this machine interrupts the execution of instructions when executing in user mode. When the timer goes to zero, the internal state bit (ISB) is triggered and goes to timeout. The timeout interrupt exception only occurs when the ISB is high and the control unit is between instructions (executed but yet to fetch a new instruction) in user mode. The ISB is not reset until a new value is loaded into the clock which is done by the CLK instruction in privileged mode. If execute is being run in privileged mode, the countdown timer has no effect.

IV. State Tables and Equations

The finite state machine is implemented in the control unit via the equations and tables in this section of the report.

A. State Diagram

The state diagram for this machine is implemented using the given design requirements. The complete state diagram that also displays each of the substates is in the **Appendix**.

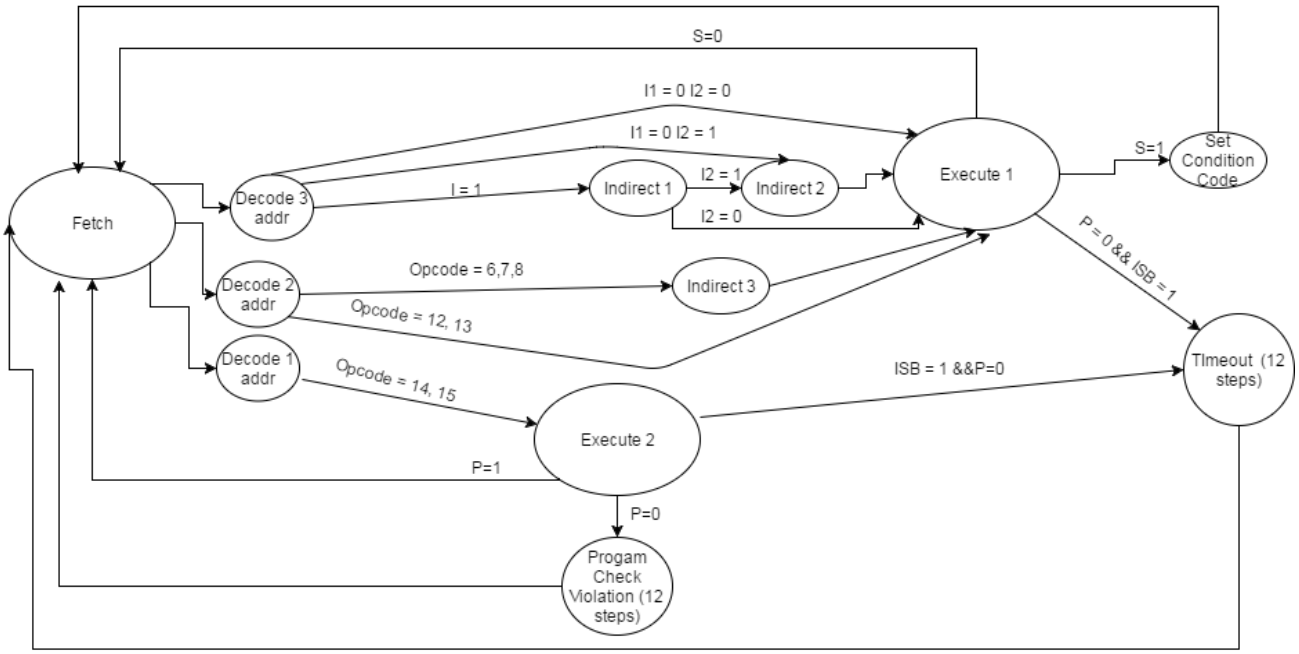

Figure 22: High Level State Diagram

B. State Addresses

Each of the instruction states in the diagram are assigned a binary address.

Fetch	**0000**
Decode 3 Address	**0001**
Decode 2 Address	**0010**
Decode 1 Address	**0011**
Indirect 1	**0100**
Indirect 2	**0101**
Indirect 3	**0110**
Execute 1	**0111**
Execute 2	**1000**
Set Condition Code	**1001**
Timeout	**1010**
Program Check Violation	**1011**

Table 5: State Addresses

C. State Equations
(written in terms of high level states and state bits Q3Q2Q1Q0)

These equations were derived from the next state table and state diagram.

Qbits(MSB->LSB)

0000 = Fetch = (Execute1*!ISB*!S) + setConditionCode +Execute2+PgmCk

0001 = Decode3addr = fetch*(3addr-> opcode 0,1,2,3,4,5)

0010 = Decode2addr = fetch*(2addr ->opcode 6,7,8,12,13)

0011 = Decode1addr = fetch*(1addr-> opcode 14,15)

0100 = Indirect1= Decode3addr*I1

0101 = Indirect2= (Indirect1*I2)+(!I1*I2)

0110 = Indirect3= (Decode2addr*(opcode = 6,7,8)

0111 = Execute1 = (Ind1*I2)+(Decode3addr*!I1+!I2)+(Indirect3)+(Decode2addr*(opcode = 12,13)

1000 = Execute 2= Decode1addr*(P+(P+opcode!=14,15)

1001 = Set.CC = Execute1*!Timeout*S

1010 = Timeout = (Execute1+Execute2)*ISB

1011 = PgmCk = (Execute2*!P)

0000 =[(Q3'Q2Q1Q0)*(!ISB)*(!S)]+ (Q3Q2'Q1'Q0)+ (Q3Q2'Q1'Q0')+(Q3Q2'Q1Q0)

0001 = (Q3'Q2'Q1'Q0')*(3 addr opcode 0,1,2,3,4,5)

0010 = (Q3'Q2'Q1'Q0')*(2 addr opcode 6,7,8,12,13)

0011 = (Q3'Q2'Q1'Q0')*(1 addr opcode 14,15)

0100 = (Q3'Q2'Q1'Q0)*I1

0101 = (Q3'Q2Q1'Q0'*I2)+(!I1*I2)

0110 = (Q3'Q2'Q1Q0'*(opcode = 6,7,8)

0111 = (Q3'Q2Q1'Q0'*I2)+(Q3'Q2'Q1'Q0*!I1+!I2)+(Q3'Q2Q1Q0')+(Q3'Q2'Q1Q0'*(opcode=12,13)

1000 = (Q3'Q2'Q1Q0*(P+(P+opcode!=14,15)

1001 = Q3'Q2Q1Q0*!(Q3Q2'Q1Q0')*S

1010 = (Q3'Q2Q1Q0+Q3Q2'Q1'Q0')*ISB

1011 = (Q3Q2'Q1'Q0'*!P)

D. Next State Table

Using the state diagram and state addresses of each instruction, a state table is created that shows the current state the machine is in, the inputs to the machine, and then the next state that instruction goes to. The next state table is shown in **Table 6**. The color code represents the current state it is in. The entire next state table that includes each of the substates is **Table 8** in the **Appendix**.

Current State				Inputs												Next State			
Q3	Q2	Q1	Q0	I1	I2	Op 6	Op 7	Op 8	Op 12	Op 13	Op 14	Op 15	PSW.P	S	ISB	Q3*	Q2*	Q1*	Q0*
0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	1
0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	0	0	1	0
0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	0	0	1	1
0	0	0	1	0	0	x	x	x	x	x	x	x	x	x	x	0	1	1	1
0	0	0	1	1	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0
0	0	1	0	x	x	1	x	x	x	x	x	x	x	x	x	0	1	1	0
0	0	1	0	x	x	x	1	x	x	x	x	x	x	x	x	0	1	1	0
0	0	1	0	x	x	x	x	1	x	x	x	x	x	x	x	0	1	1	0
0	0	1	0	x	x	x	x	x	1	x	x	x	x	x	x	0	1	1	1
0	0	1	0	x	x	x	x	x	x	1	x	x	x	x	x	0	1	1	1
0	0	1	1	x	x	x	x	x	x	x	1	x	x	x	x	1	0	0	0
0	0	1	1	x	x	x	x	x	x	x	x	1	x	x	x	1	0	0	0
0	1	0	0	x	0	x	x	x	x	x	x	x	x	x	x	0	1	1	1
0	1	0	0	x	1	x	x	x	x	x	x	x	x	x	x	0	1	0	1
0	1	0	1	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1
0	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1
0	1	1	1	x	x	x	x	x	x	x	x	x	x	1	x	1	0	0	1
0	1	1	1	x	x	x	x	x	x	x	x	x	0	x	1	1	0	1	0
0	1	1	1	x	x	x	x	x	x	x	x	x	x	0	x	0	0	0	0
1	0	0	0	x	x	x	x	x	x	x	x	x	0	x	x	1	0	1	1
1	0	0	0	x	x	x	x	x	x	x	x	x	1	x	x	0	0	0	0
1	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0
1	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0
1	0	1	1	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0

Table 6: Next State Table

V. Control Unit

A. Explanation

The control unit for this machine is implemented by a finite state machine connected to a sequencer that iterates through each substate. The finite state machine progresses through each state based on sequential logic of the previous states as well as on inputs to the machine. These inputs include the opcodes with assorted control bits (I1, I2, S), privileged/user bits, and the internal state bit for the timeout. As the finite state machine progresses through each state, it must also cycle through the requisite number of substates depending on what operation it is performing. For example, the fetch cycle has three substates. When the fetch state is active, the sequencer will iterate three times before it enables the finite state machine to progress to the next state. During each iteration, the sequencer generates control signals which are input to a 4-1 multiplexer. The multiplexer is set to propagate these signals to tri state buffers connected to each datapath element on the data path to either enable or disable said component.

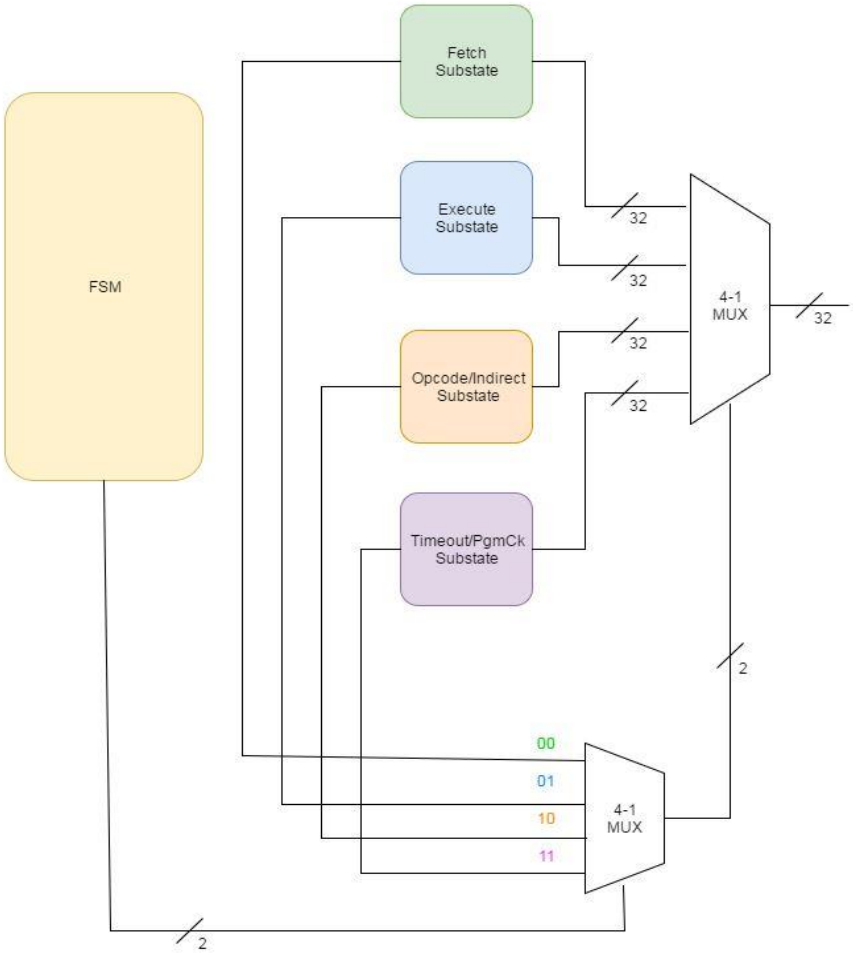

Figure 23: Control Unit Block Diagram

B. Finite State Machine Implementation

The finite state machine without substate sequencing is implemented as shown below.

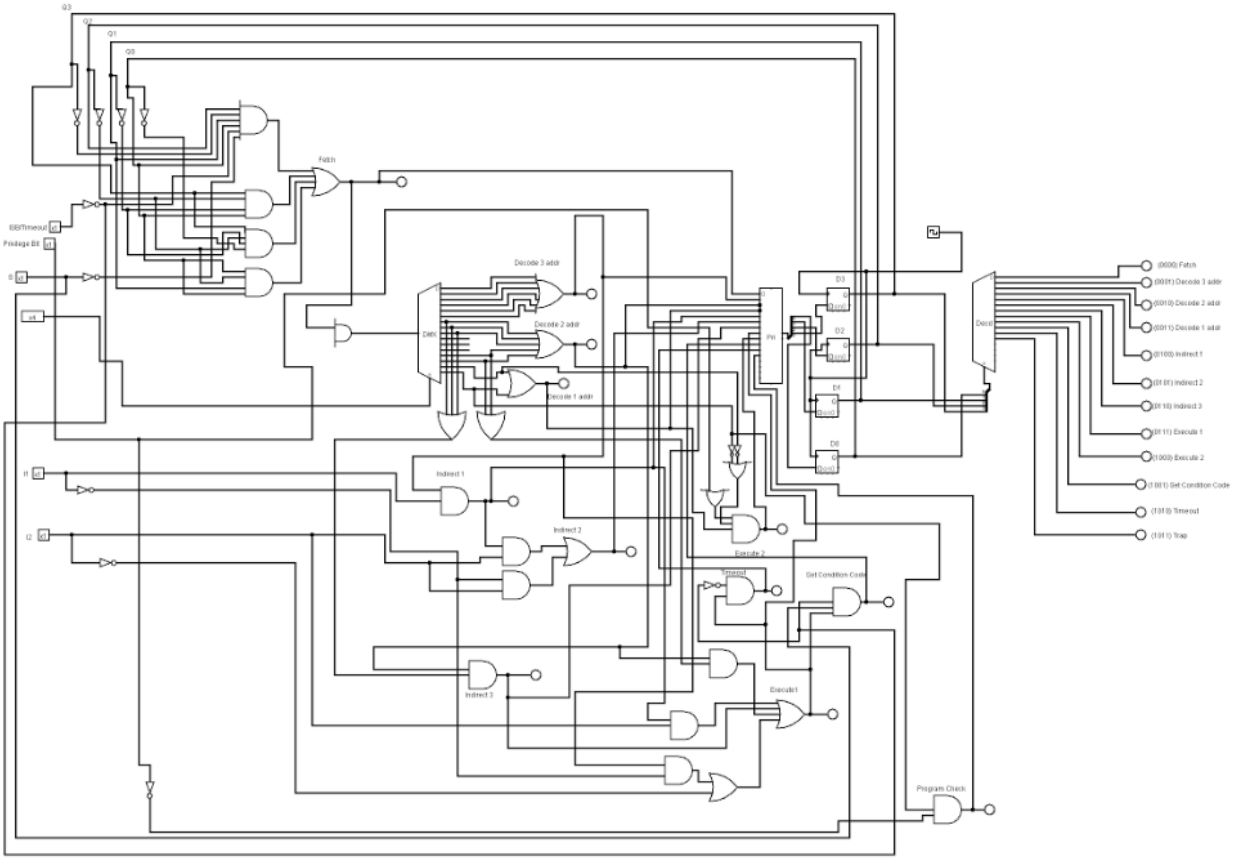

Figure 24: Finite State Machine without Substate Sequencer

C. Substate Sequencing Explanation and Implementation

The entire substate sequencer is shown in the **Appendix**. Diagrams of each substate sequencer block are shown below (a total of four). They will be input to the 4-1 multiplexer are as follows: 00 = fetch, 01 = execute, 10 = decode/indirect, 11 = timeout/program check violation. The execute block is split into five more blocks for readability, however in the machine it is still implemented as one substate block. Each sequencer is labeled with the generated control signals that are enabled. Every control signal not enabled is not shown for readability purposes. One complete example will be shown in section **VI.I**.

Each substate is implemented with combinational logic. When the super state is active, it enables the substate. The clock then counts up through the requisite number of substate iterations via a counter connected to a decoder. The FSM will not iterate to the next state until the substate is complete because the clock is connected back to the FSM via an and gate that is connected to an or gate connecting all of the “finished substate” outputs. This restricts the FSM to progressing to the next state only if the current substate is complete because the clock input will not propagate otherwise. The “finished substate” wire can be seen in all of the substate blocks, but the implementation will still be shown for documentation purposes.

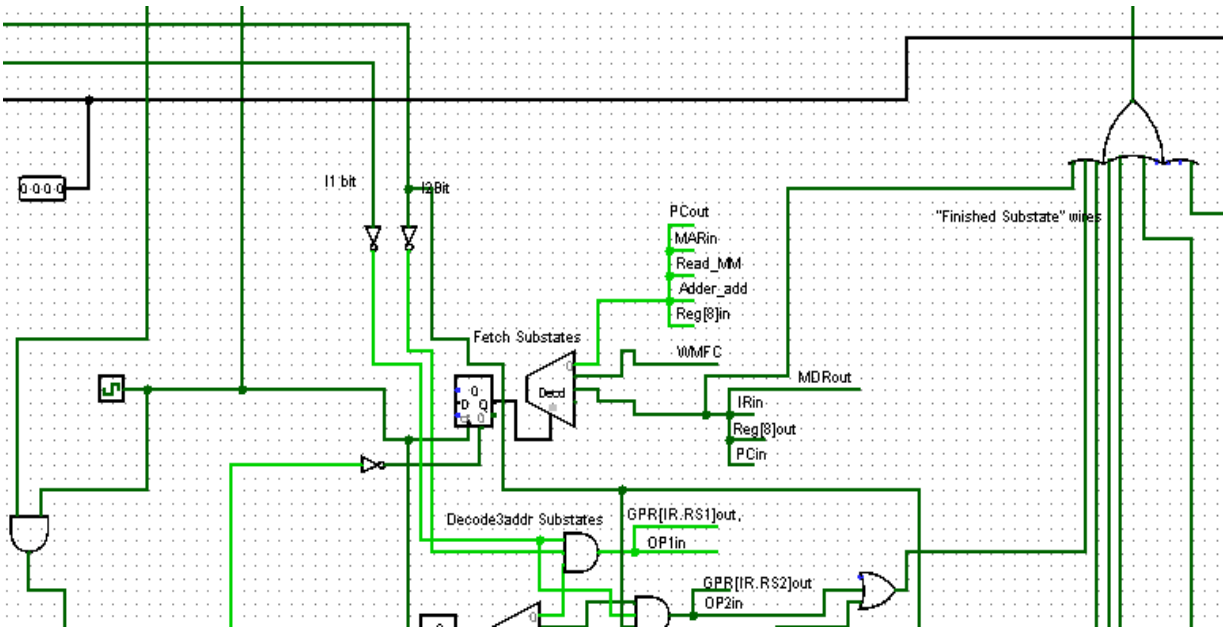

Figure 25: Clock to FSM
(enabled by finished substate on left, finished substate signal on right)

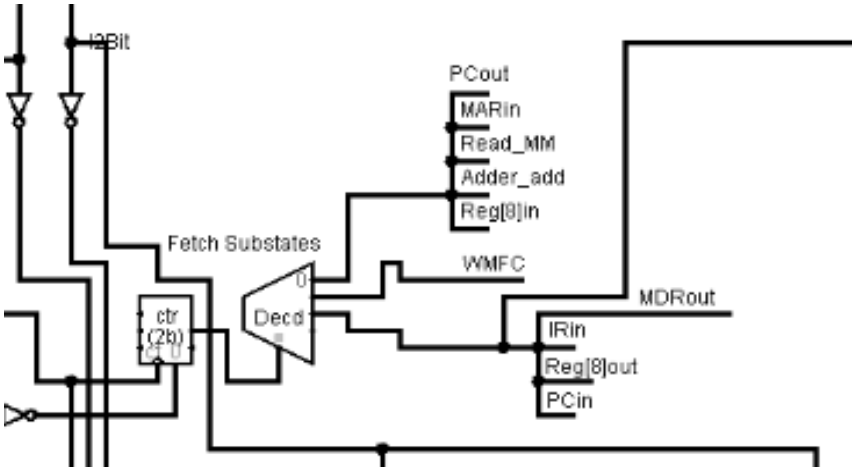

Figure 26: Fetch Substate Block

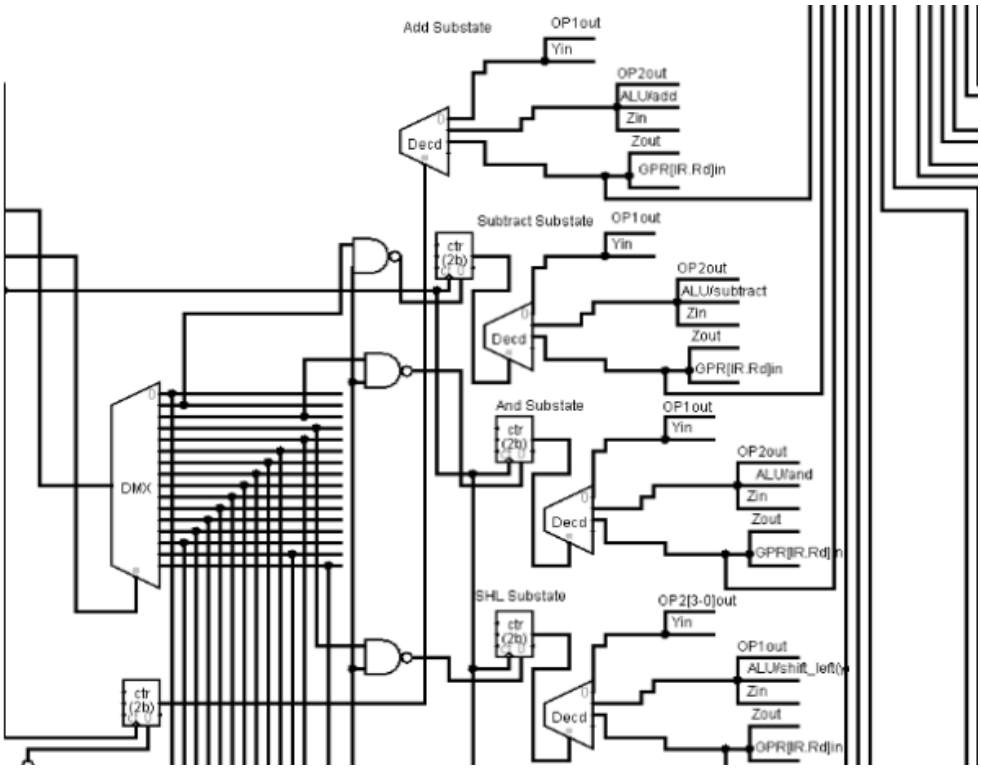

Figure 27: Execute Substate Block 1 (Opcodes 0-3)

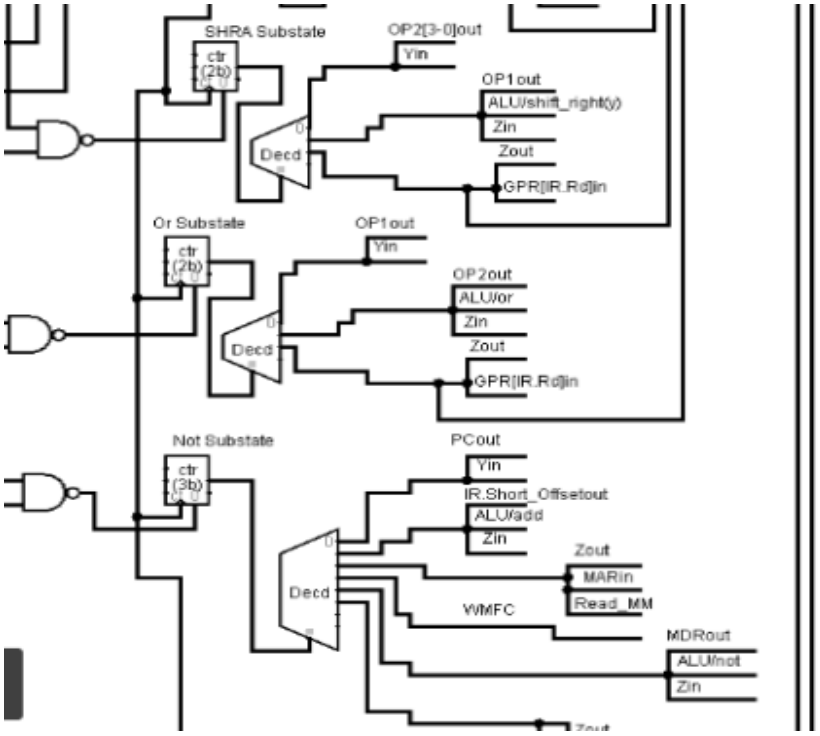

Figure 28: Execute Substate Block 2 (Opcodes 4-6)

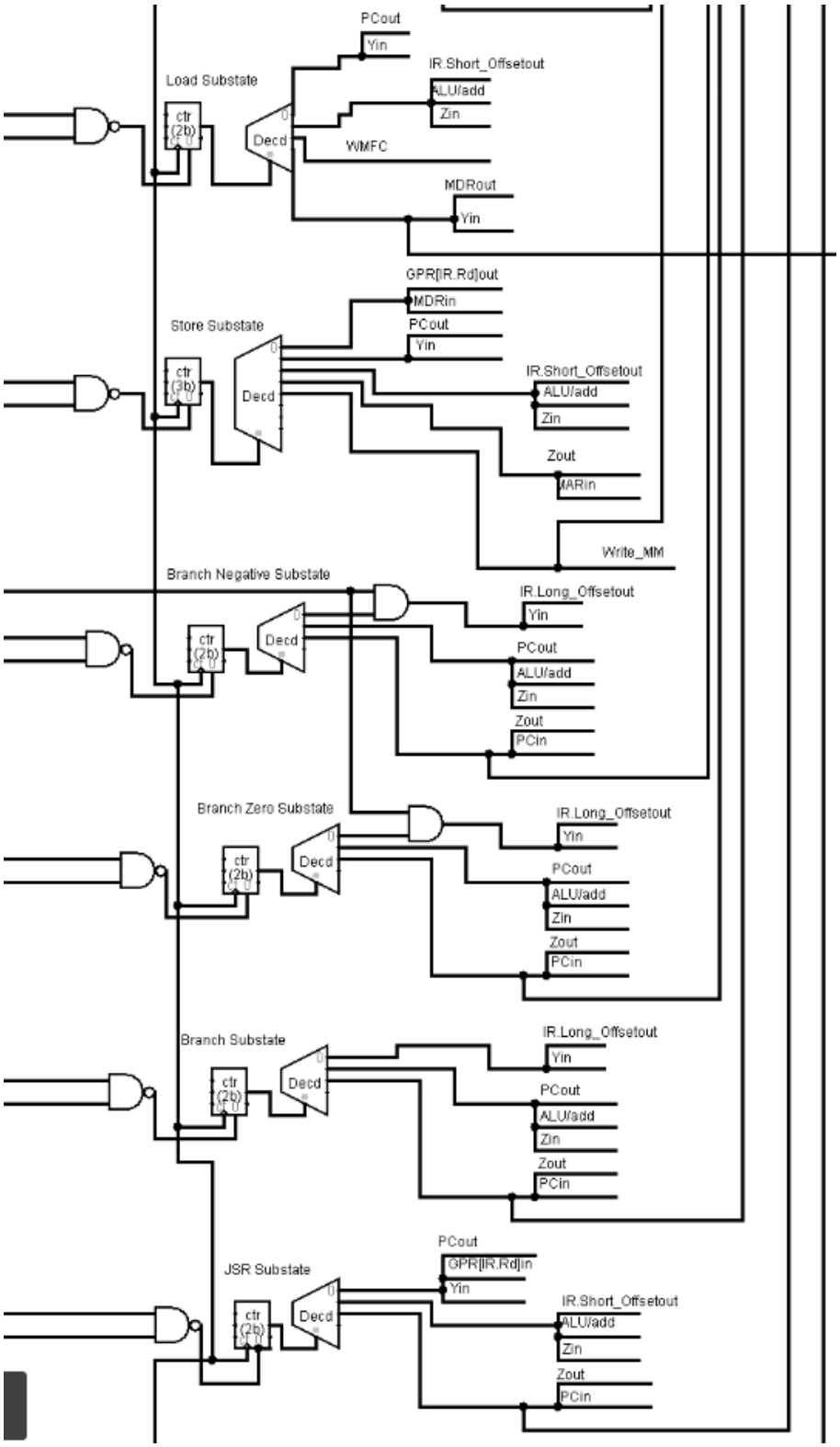

Figure 29: Execute Substate Block 3 (Opcodes 7-12)

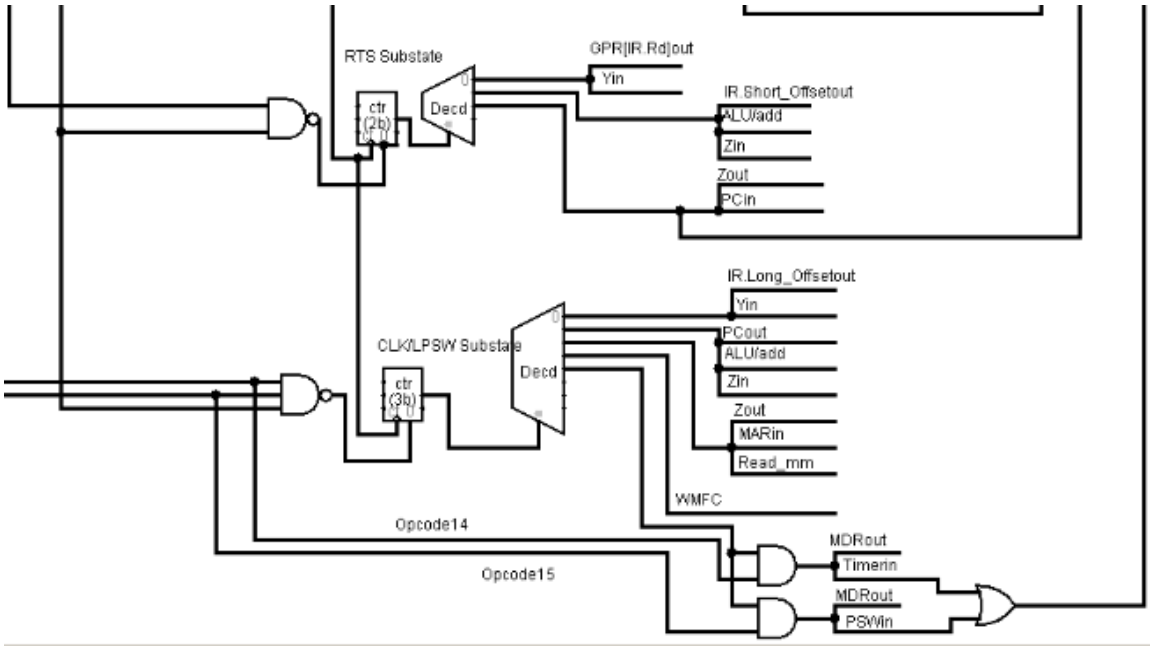

Figure 30: Execute Substate Block 4 (Opcodes 13-15)

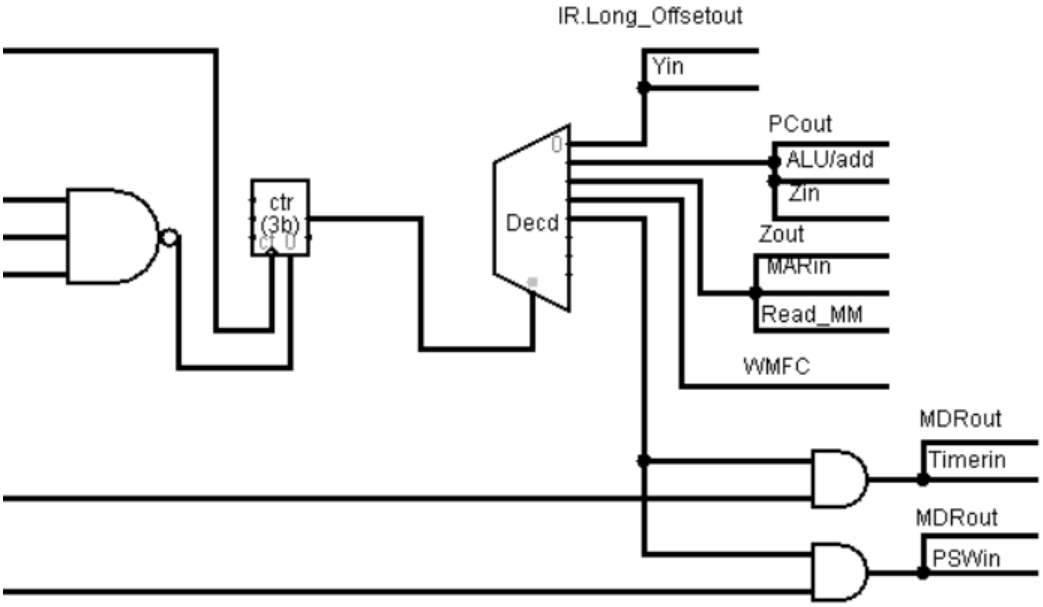

Figure 31: Execute Substate Block 5 (Execute 2)

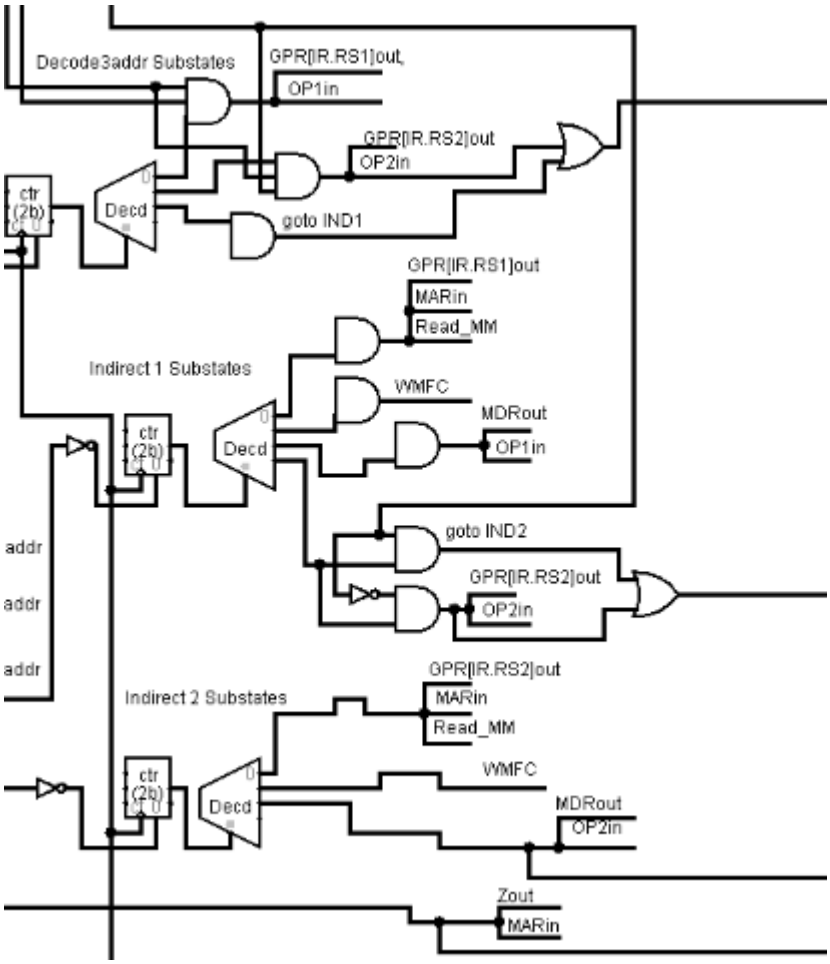

Figure 32: Decode/Indirect Substate Block

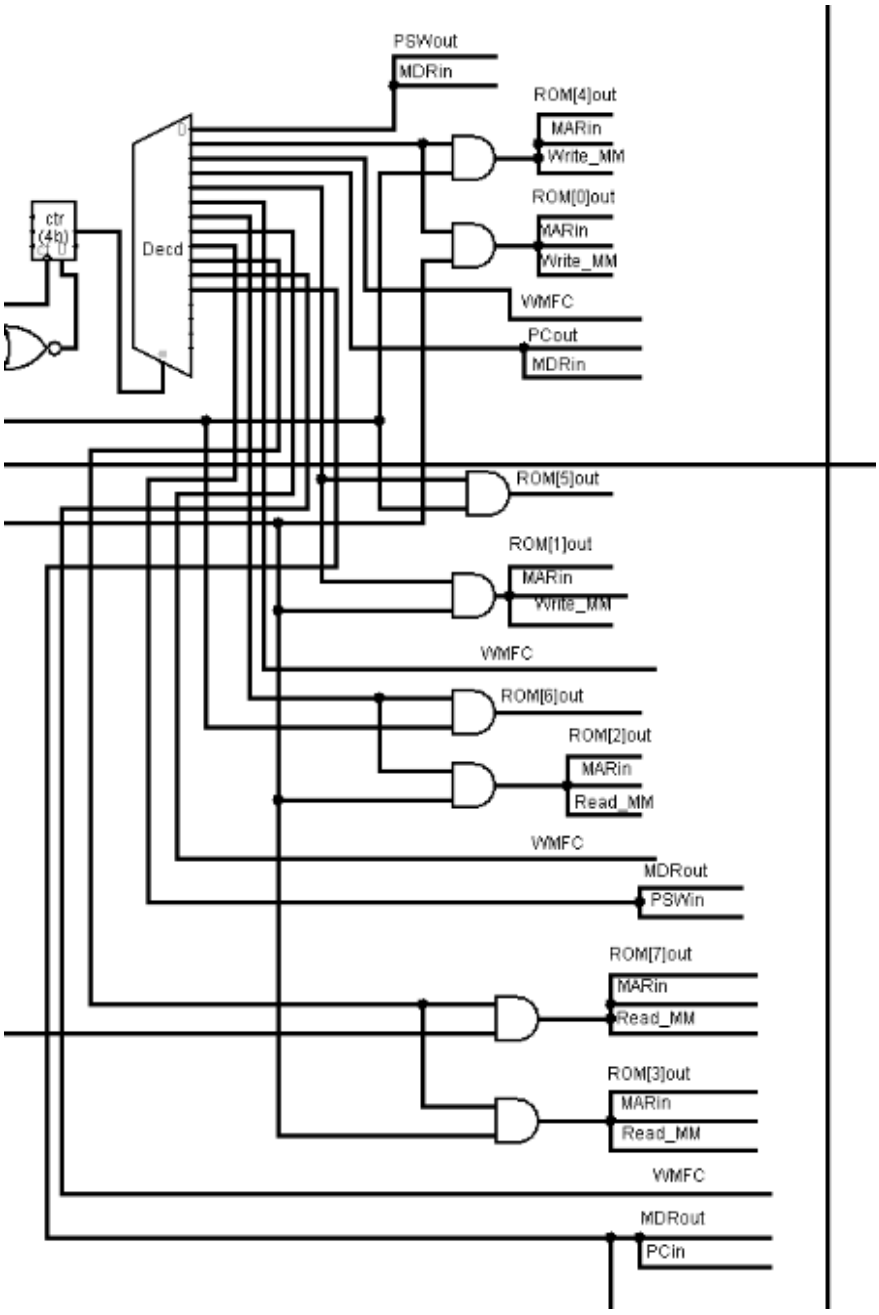

Figure 33: Timeout/Program Check Violation Substate Block

D. Control Signal Explanation

There are a total of 30 different control signals for this machine. Each substate block (fetch, decode/indirect, execute, timeout/program check) has a 32 bit output from each substate iteration to the 4-1 control signal multiplexer. These 32 bits create a control string which either enable or disable registers and memory elements depending on their value (0= disable, 1 = enable). For GPRin/out and ROMout functions, the address is specified by combinational logic in the substate sequencer.

E. Control Signal List

The complete list of control signals and their functionality is shown in the table below. Bits 31 and 32 are denoted as x (don't care) because no control signal is assigned to them.

Control Signal Name	Control Signal Bit Number (0-31) (Least sig-Most sig bit)	Control Signal Functionality
PCout	10000000000000000000000000000xx	Outputs the contents of the Program Counter (GPR Register 7) to the data bus.
PCin	01000000000000000000000000000xx	Inputs data currently on the bus into the Program Counter (GPR Register 7)
MARin	00100000000000000000000000000xx	Inputs data from the bus to the Memory Address Register, which is then used to locate the corresponding address in main memory
Read_MM	00010000000000000000000000000xx	Reads the data stored in main memory at the address pointed to by the MAR
Write_MM	00001000000000000000000000000xx	Writes data to be stored in main memory at the address pointed to by the MAR
WMFC	00000100000000000000000000000xx	Wait for Memory Function to Complete - when enabled, sends 0/x to every other control bit in the string so that the machine waits for one clock cycle while memory functions are in progress
MDRout	00000010000000000000000000000xx	Outputs the contents of the Memory Data Register onto the bus
MDRin	00000001000000000000000000000xx	Inputs the data from main memory into the MDR
IRin	00000000100000000000000000000xx	Inputs data from the bus into the Instruction Register

Reg[8]out	00000000010000000000000000000xx	Outputs data from Register 8 in the register file to the PC for autoincrementing purposes
Reg[8]in	00000000001000000000000000000xx	Inputs the incremented PC from the adder in the register file into Register 8
OP1out	00000000000100000000000000000xx	Outputs data from the OP1 register to the bus
OP1in	00000000000010000000000000000xx	Inputs data from the bus into the OP1 register
OP2out	00000000000001000000000000000xx	Outputs data from the OP2 register to the bus
OP2in	00000000000000100000000000000xx	Inputs data from the bus into the OP2 register
Yin	00000000000000010000000000000xx	Inputs data from the bus to the Yin register, to be used in ALU operations (automatically propagated to ALU input b)
Zin	00000000000000001000000000000xx	Inputs the result from ALU operations into the Z register
Zout	00000000000000000100000000000xx	Outputs the contents of the Z register onto the bus
GPRin	00000000000000000010000000000xx	Inputs data from the bus to open registers in the data file, permitted they are registers 2-6. Registers 1 and 7 are reserved. The register is specified by the opcode, which can specify Rs1 (register source 1), Rs2 (register source 2), Rd (register destination) for 3 address operations, and Rd for 2 address operations
GPRout	00000000000000000001000000000	Outputs data from the register file

	00xx	registers 2-6 onto the bus
OP2[3-0]out	00000000000000000000010000000xx	Outputs the 4 least significant bits of the contents of register OP2 onto the bus
IR.Short_Offset out	00000000000000000000100000000xx	Shifts the PC by 4 bits for 2 address opcodes
IR.Long_Offset out	00000000000000000000010000000xx	Shifts the PC by 8 bits for 1 address opcodes
PSWout	00000000000000000000001000000xx	Outputs the contents of the PSW (Program Status Word) register onto the bus
ROM[out]	00000000000000000000000100000xx	Outputs the constant value stored in the ROM specified by the control string and combinational logic
ROM[in]	00000000000000000000000010000xx	Does nothing (read only memory)
PSWin	00000000000000000000000001000xx	Inputs data from the bus into the PSW register
adder_add	00000000000000000000000001000xx	Enables the adder in the register file to add 2 to the PC
ALU/OP	00000000000000000000000000001000xx	Enables the ALU control unit - ALU controls specified in the ALU control unit section of the report (a total of 8 different operations)
Timerin/CLKin	00000000000000000000000000010xx	Inputs data from the bus into the Clock, which is connected to the timer

F. Machine Operations with Control Signals (separated into substates in which they occur)

For notation purposes the necessary ALU operation (add/subract, etc) performed is shown in parentheses next to the ALU/op signal.

Fetch (0000)
1. PC_{out} , MAR_{in}, read_MM, adder_add, $Reg8_{in}$
2. WMFC
3. MDR_{out}, IR_{in}, $Reg8_{out}$, PC_{in}

3 address instructions

ADD opcode 0 operation: GPR[Rd]=OP1+OP2
1. $OP1_{out-}$, Y_{in}
2. $OP2_{out}$, ALU/op(add), Z_{in-}
3. Z_{out}, GPR_{in}

SUB opcode 1 operation: GPR[Rd]=OP1-OP2
1. $OP1_{out}$, Y_{in}
2. $OP2_{out}$, ALU/op(subtract), Z_{in}
3. Z_{out}, GPR_{in}

AND opcode 2 operation: GPR[Rd]=OP1 and OP2
1. $OP1_{out}$, Y_{in}
2. $OP2_{out}$, ALU/op(and), Z_{in}
3. Z_{out}, GPR_{in-}

SHL opcode 3 operation: GPR[Rd]=shift_left(OP1) by $OP2_{3-0}$
1. $OP2[3-0]_{out}$, Y_{in}
2. $OP1_{out}$, ALU/op(shll), Z_{in}
3. Z_{out}, GPR_{in}

SHRA opcode 4 operation: GPR[Rd]=shift_right(OP1) by $OP2_{3-0}$
1. $OP2[3-0]_{out}$, Y_{in}
2. $OP1_{out}$, ALU/op(shra), Z_{in}
3. Z_{out}, GPR_{in}

OR opcode 5 operation: GPR[Rd] = OP1 or OP2
1. $OP1_{out}$, Y_{in-}
2. $OP2_{out}$, ALU/op(or), Z_{in}
3. Z_{out}, GPR_{in}

2 address instructions

NOT opcode 6 operation: GPR[Rd] = not MM[PC + Short_Offset]

1. PC_{out}, Y_{in}
2. $IR.Short\_Offset_{out}$, ALU/op(add), Z_{in}
3. Z_{out}, MAR_{in}, read_MM
4. WMFC
5. MDR_{out}, ALU/op(not), Z_{in}
6. Z_{out}, GPR_{in}

LD opcode 7 operation: GPR[Rd] = MM[PC + Short_Offset]

1. PC_{out}, Y_{in}
2. $IR.Short\_Offset_{out}$, ALU/op(add), Z_{in}
3. Z_{out}, MAR_{in}, read_MM
4. WMFC
5. MDR_{out}, GPR_{in}

ST opcode 8 operation: MM[PC + Short_Offset] = GPR[Rd]

1. GPR_{out}, MDR_{in}
2. PC_{out}, Y_{in}
3. [$IR.Short\_Offset_{out}$, ALU/op(add), Z_{in}
4. Z_{out}, MAR_{in}
5. Write_MM

JSR opcode 12 operation: GPR[Rd] = PC; PC = PC + Short_Offset

1. PC_{out}, GPR_{in}, Y_{in}
2. IR.Short_Offset, ALU/op(add), Z_{in}
3. Z_{out}, PC_{in}

RTS opcode 13 operation: PC = GPR[Rd] + Short_Offset

1. GPR_{out}, $Y\text{-}_{in}$
2. IR.Short_Offset, ALU/op(add), Z_{in}
3. Z_{out}, PC_{in}

1 address instructions

BRN opcode 9 operation: if CC.N then PC = PC + Long_Offset

1. If CC.N = 1
   IR.Long_Offset$_{out}$, Y$_{in}$
2. PC$_{out}$, ALU/op(add), Z$_{in}$
3. Z$_{out}$, PC$_{in}$

BRZ opcode 10 operation: if CC.Z then PC = PC + Long_Offset

1. If CC.Z = 1
   IR.Long_Offset$_{out}$, Y$_{in}$
2. PC$_{out}$, ALU/op(add), Z$_{in}$
3. Z$_{out}$, PC$_{in}$

BR opcode 11 operation: PC = PC + Long_Offset

1. IR.Long_Offset$_{out}$, Y$_{in}$
2. PC$_{out}$, ALU/op(add), Z$_{in}$
3. Z$_{out}$, PC$_{in}$

CLK opcode 14 operation: set timer to MM[PC + Long_Offset]

1. IR.Long_Offset$_{out}$, Y$_{in}$
2. PC$_{out}$, ALU/op(add), Z$_{in}$
3. Z$_{out}$, MAR$_{in}$, Read_MM
4. WMFC
5. MDR$_{out}$, Timer$_{in}$

LPSW opcode 15, operation: PSW = MM[PC + Long_Offset]

1. IR.Long_Offset$_{out}$, Y$_{in}$
2. PC$_{out}$, ALU/op(add), Z$_{in}$
3. Z$_{out}$, MAR$_{in}$, Read_MM
4. WMFC
5. MDR$_{out}$, PSW$_{in}$

Decode3addr (0001)
1. GPR_{out}, $OP1_{in}$
2. If I2=0
   GPR_{out},$OP2_{in}$
   Else if I2 = 1 goto Indirect2 (0101)

Decode2addr (0010)
1. $IR.Short\_offset_{out}$, ALU/op(add), Z_{in}

Decode1addr (0011)
1. $IR.Long\_offset_{out}$, ALU/op(add), Z_{in}

Indirect1 (0100)
1. GPR_{out}, MAR_{in}, Read_MM
2. WMFC
3. MDR_{out},$OP1_{in}$,
4. WMFC
5. MDR_{out}, PSW_{in}

Indirect2 (0101)
1. GPR_{out}, MAR_{in}, Read_MM
2. WMFC
3. MDR_{out},$OP2_{in}$,
4. WMFC

Indirect3 (0110)
1. Z_{out}, MAR_{in}

Timeout (1010)
1. PSW_{out}, MDR_{in}
2. ROM_{out}, MAR_{in}, Write_MM
3. WMFC
4. PC_{out}, MDR_{in}
5. ROM_{out}
6. WMFC
7. ROM_{out}
8. WMFC
9. MDR_{out}
10. ROM_{out}, MAR_{in}, Read_MM
11. WMFC
12. MDR_{out}, PC_{in}

Program Check Violation/Trap (1011)

1. $PSW_{out,}$ MDR_{in}
2. $ROM_{out,}$ $MAR_{in,}$ Write_MM
3. WMFC
4. $PC_{out,}$ MDR_{in}
5. $ROM_{out,}$ MAR_{in} Write_MM
6. WMFC
7. $ROM_{out,}$ MAR_{in} Read_MM
8. WMFC
9. $MDR_{in,}$ PSW_{in}
10. $ROM_{out,}$ $MAR_{in,}$ Read_MM
11. WMFC
12. $MDR_{out,}$ PC_{in}

G. Control String Derivation

	Stages/Substate Steps											
FETCH (0000)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	1	0	0									
PCin	0	0	1									
MARin	1	0	0									
Read_MM	1	0	0									
Write_MM	0	0	0									
WMFC	0	1	0									
MDRout	0	0	1									
MDRin	0	0	0									
IRin	0	0	1									
Reg[8]out	0	0	1									
Reg[8]in	1	0	0									
OP1out	x	x	x									
OP1in	x	x	x									
OP2out	x	x	x									
OP2in	x	x	x									
Yin	x	x	x									
Zin	x	x	x									
Zout	x	x	x									
GPRin	x	x	x									
GPRout	x	x	x									
OP2[3-0]out	x	x	x									
IR.Short_Offsetout	x	x	x									
IR.Long_Offsetout	x	x	x									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	1	0	0									
ALU/OP	x	x	x									
Timerin/CLKin	x	x	x									

Table 7: Fetch Control Signal Derivation Table

The control strings for each operation listed in **VI.F** were derived using the table shown in **Table 7** above. The table for the fetch cycle is shown here, and the rest of the tables are shown in the **Appendix.**

H. Control String List

Each control string is listed from the least significant bit to the most significant bit (0-31) which correspond to the control signal list in section **VI.E**
The control strings are listed in the order they execute. For notation purposes, X's denote control signals that are unused in the operation. In reality they are 0's (except for the unused 30 and 31 bits)

Fetch

1 0 1 1 0 0 0 0 0 0 1 x x x x x x x x x x x x x x x x 1 x x x x

0 0 0 0 0 1 0 1 0 0 0 x x x x x x x x x x x x x x x x 0 x x x x

0 1 0 0 0 0 1 0 1 1 0 x x x x x x x x x x x x x x x x 0 x x x x

Add (Opcode 0)

x x x x x x x x x x 1 x 0 x 1 0 0 x x x x x x x x x x 0 x x x

x x x x x x x x x x 0 x 1 x 0 1 0 x x x x x x x x x x 1 x x x

x x x x x x x x x x 0 x 0 x 0 0 1 x x x x x x x x x x 0 x x x

Subtract (Opcode 1)

x x x x x x x x x x 1 x 0 x 1 0 0 x x x x x x x x x x 0 x x x

x x x x x x x x x x 0 x 1 x 0 1 0 x x x x x x x x x x 1 x x x

x x x x x x x x x x 0 x 0 x 0 0 1 x x x x x x x x x x 0 x x x

And (Opcode 2)

x x x x x x x x x x 1 x 0 x 1 0 0 x x x x x x x x x x 0 x x x

x x x x x x x x x x 0 x 1 x 0 1 0 x x x x x x x x x x 1 x x x

x x x x x x x x x x 0 x 0 x 0 0 1 x x x x x x x x x x 0 x x x

SHL (Opcode 3)

x x x x x x x x x x 0 x 1 0 0 0 x 1 x x x x x x 0 x x x

x x x x x x x x x x 1 x 0 1 0 0 x 0 x x x x x x 1 x x x

x x x x x x x x x x 0 x 0 0 1 1 x 0 x x x x x x 0 x x x

SHRA (Opcode 4)

x x x x x x x x x x x x x 0 x 1 0 0 0 x 1 x x x x x x x 0 x x x

x x x x x x x x x x x x x 1 x 0 1 0 0 x 0 x x x x x x x 1 x x x

x x x x x x x x x x x x x 0 x 0 0 1 1 x 0 x x x x x x x 0 x x x

OR (Opcode 5)

x x x x x x x x x x x 1 x 0 x 1 0 0 0 x x x x x x x x x 0 x x x

x x x x x x x x x x x 0 x 1 x 0 1 0 0 x x x x x x x x x 1 x x x

x x x x x x x x x x x 0 x 0 x 0 0 1 1 x x x x x x x x x 0 x x x

Not (Opcode 6)

1 x 0 0 x 0 0 x x x x x x x x 1 0 0 0 x x 0 x x x x x x 0 x x x

0 x 0 0 x 0 0 x x x x x x x x 0 1 0 0 x x 1 x x x x x x 1 x x x

0 x 1 1 x 0 0 x x x x x x x x 0 0 1 0 x x 0 x x x x x x 0 x x x

0 x 0 0 x 1 0 x x x x x x x x 0 0 0 0 x x 0 x x x x x x 0 x x x

0 x 0 0 x 0 1 x x x x x x x x 0 1 0 0 x x 0 x x x x x x 1 x x x

0 x 0 0 x 0 0 x x x x x x x x 0 0 1 1 x x 0 x x x x x x 0 x x x

Load (Opcode 7)

1 x 0 0 x 0 0 x x x x x x x x 1 0 0 0 x x 0 x x x x x x 0 x x x

0 x 0 0 x 0 0 x x x x x x x x 0 1 0 0 x x 1 x x x x x x 1 x x x

0 x 1 1 x 0 0 x x x x x x x x 0 0 1 0 x x 0 x x x x x x 0 x x x

0 x 0 0 x 1 0 x x x x x x x x 0 0 0 0 x x 0 x x x x x x 0 x x x

0 x 0 0 x 0 1 x x x x x x x x 0 0 0 1 x x 0 x x x x x x 0 x x x

Store (Opcode 8)

0 x 0 x 0 x x 1 x x x x x x x x 0 0 0 x 1 x 0 x x x x x x 0 x x x

1 x 0 x 0 x x 0 x x x x x x x 1 0 0 x 0 x 0 x x x x x x 0 x x x

0 x 0 x 0 x x 0 x x x x x x x 0 1 0 x 0 x 1 x x x x x x 1 x x x

0 x 1 x 0 x x 0 x x x x x x x 0 0 1 x 0 x 0 x x x x x x 0 x x x

0 x 0 x 1 x x 0 x x x x x x x 0 0 0 x 0 x 0 x x x x x x 0 x x x

Branch Negative (Opcode 9)

0 0 x x x x x x x x x x x x x 1 0 x x x x x 1 x x x x x 0 x x x

1 0 x x x x x x x x x x x x x 0 1 x x x x x 0 x x x x x 1 x x x

0 1 x x x x x x x x x x x x x 0 1 x x x x x 0 x x x x x 0 x x x

Branch Zero (Opcode 10)

0 0 x x x x x x x x x x x x x 1 0 x x x x x 1 x x x x x 0 x x x

1 0 x x x x x x x x x x x x x 0 1 x x x x x 0 x x x x x 1 x x x

0 1 x x x x x x x x x x x x x 0 1 x x x x x 0 x x x x x 0 x x x

Branch (Opcode 11)

0 x x x x x x x x x x x x x 1 0 0 x x x x 1 x x x x x 0 x x x

1 x x x x x x x x x x x x x 0 1 0 x x x x 0 x x x x x 1 x x x

1 x x x x x x x x x x x x x 0 0 1 x x x x 0 x x x x x 0 x x x

JSR (Opcode 12)

1 0 x x x x x x x x x x x x x 1 0 0 1 x x 0 x x x x x x 0 x x x

0 0 x x x x x x x x x x x x x 0 1 0 0 x x 1 x x x x x x 1 x x x

0 1 x x x x x x x x x x x x x 0 0 1 0 x x 0 x x x x x x 0 x x x

RTS (Opcode 13)

x 0 x x x x x x x x x x x x x 1 0 0 x 1 x 0 x x x x x x 0 x x x

x 0 x x x x x x x x x x x x x 0 1 0 x 0 x 1 x x x x x x 1 x x x

x 1 x x x x x x x x x x x x x 0 0 1 x 0 x 0 x x x x x x 0 x x x

CLK (Opcode 14)

0 x 0 0 x 0 0 x x x x x x x x 1 0 0 x x x 1 x x x x x x 0 0 x x

1 x 0 0 x 0 0 x x x x x x x x 0 1 0 x x x 0 x x x x x x 1 0 x x

0 x 1 1 x 0 0 x x x x x x x x 0 0 1 x x x 0 x x x x x x 0 0 x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x 0 x x

0 x 0 0 x 0 1 x x x x x x x x 0 0 0 x x x 0 x x x x x x 0 1 x x

LPSW (Opcode 15)

0 x 0 0 x 0 0 x x x x x x x x 1 0 0 x x x 1 x x x x 0 x 0 0 x x

1 x 0 0 x 0 0 x x x x x x x x 0 1 0 x x x 0 x x x x 0 x 1 0 x x

0 x 1 1 x 0 0 x x x x x x x x 0 0 1 x x x 0 x x x x 0 x 0 0 x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x 0 x x 0 x x

0 x 0 0 x 0 1 x x x x x x x x 0 0 0 x x x 0 x x x x 1 x 0 1 x x

Decode3addr when I = 0 (0001)

x x x x x x x x x x x x 1 x 0 x x x x 1 x x x x x x x x x x x x

x x x x x x x x x x x x 0 x 1 x x x x 1 x x x x x x x x x x x x

Decode2addr (0010)

x x x x x x x x x x x x x x x x 1 x x x x x 1 x x x x x 1 x x x

Decode1addr (0011)

x x x x x x x x x x x x x x x x 1 x x x x x 1 x x x x x 1 x x x

Indirect 1 (0100)

x x 1 1 x 0 0 x x x x x 0 x 0 x x x x 1 x x x x x x x x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

x x 0 0 x 0 1 x x x x x 1 x 0 x x x x 0 x x x x x x x x x x x x

x x 0 0 x 0 0 x x x x x 0 x 1 x x x x 1 x x x x x x x x x x x x

Indirect 2 (0101)

x x 1 1 x 0 0 x x x x x 0 x 0 x x x x 1 x x x x x x x x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

x x 0 0 x 0 1 x x x x x 1 x 0 x x x x 0 x x x x x x x x x x x x

Indirect 3 (0110)

x x 1 x x x x x x x x x x x x x 1 x x x x x x x x x x x x x

Timeout (1010)

0 0 0 0 0 0 0 1 x x x x x x x x x x x x x x x 1 0 x 0 x x x x x

0 0 1 0 1 0 0 0 x x x x x x x x x x x x x x x 0 1 x 0 x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

1 0 0 0 0 0 0 1 x x x x x x x x x x x x x x x 0 0 x 0 x x x x x

0 0 0 0 0 0 0 0 0 x x x x x x x x x x x x x x x 0 1 x 0 x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

0 0 0 0 0 0 0 0 0 x x x x x x x x x x x x x x x 0 1 x 0 x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

0 0 0 0 0 0 1 0 x x x x x x x x x x x x x x x 0 0 x 1 x x x x x

0 0 1 1 0 0 0 0 x x x x x x x x x x x x x x x 0 1 x 0 x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

0 1 0 0 0 0 1 0 x x x x x x x x x x x x x x x x x x x x x x x x

Program Check/Trap (1011)

0 0 0 0 0 0 0 1 x x x x x x x x x x x x x x x 1 0 x 0 x x x x x

0 0 1 0 1 0 0 0 x x x x x x x x x x x x x x x 0 1 x 0 x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

1 0 0 0 0 0 0 1 x x x x x x x x x x x x x x x 0 0 x 0 x x x x x

0 0 1 0 1 0 0 0 x x x x x x x x x x x x x x x 0 1 x 0 x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

0 0 1 1 0 0 0 0 x x x x x x x x x x x x x x x 0 1 x 0 x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

0 0 x x 0 0 0 1 x x x x x x x x x x x x x x x 0 0 x 1 x x x x x

0 0 1 1 0 0 0 0 x x x x x x x x x x x x x x x 0 1 x 0 x x x x x

x x x x x 1 x x x x x x x x x x x x x x x x x x x x x x x x x x

0 0 x x 0 0 1 0 x x x x x x x x x x x x x x x 0 0 x 0 x x x x x

I. Control Signal Generation

The expanded fetch substate sequencer (refer to **Figure 26 -Fetch Substate Block)** is shown in the figure on the next page. During each iteration of the substate, the necessary control signals for the current operation are enabled and the rest are disabled. Each of these control strings is sent propagated as a 32 bit input to the 4-1 control signal multiplexer. This 4-1 multiplexer is controlled by the finite state machine to dictate which substate block is active (refer to **VII.X).** As each state iterates the outputs are propagated to the multiplexer, which then enables the corresponding registers /ALU on the data path via tri state buffers. (refer to **Figure 1).**
An item to note is that during the WMFC (wait for memory function to complete) signal, all other signals are disabled, which allows the machine to wait for a clock cycle while memory cycles complete.

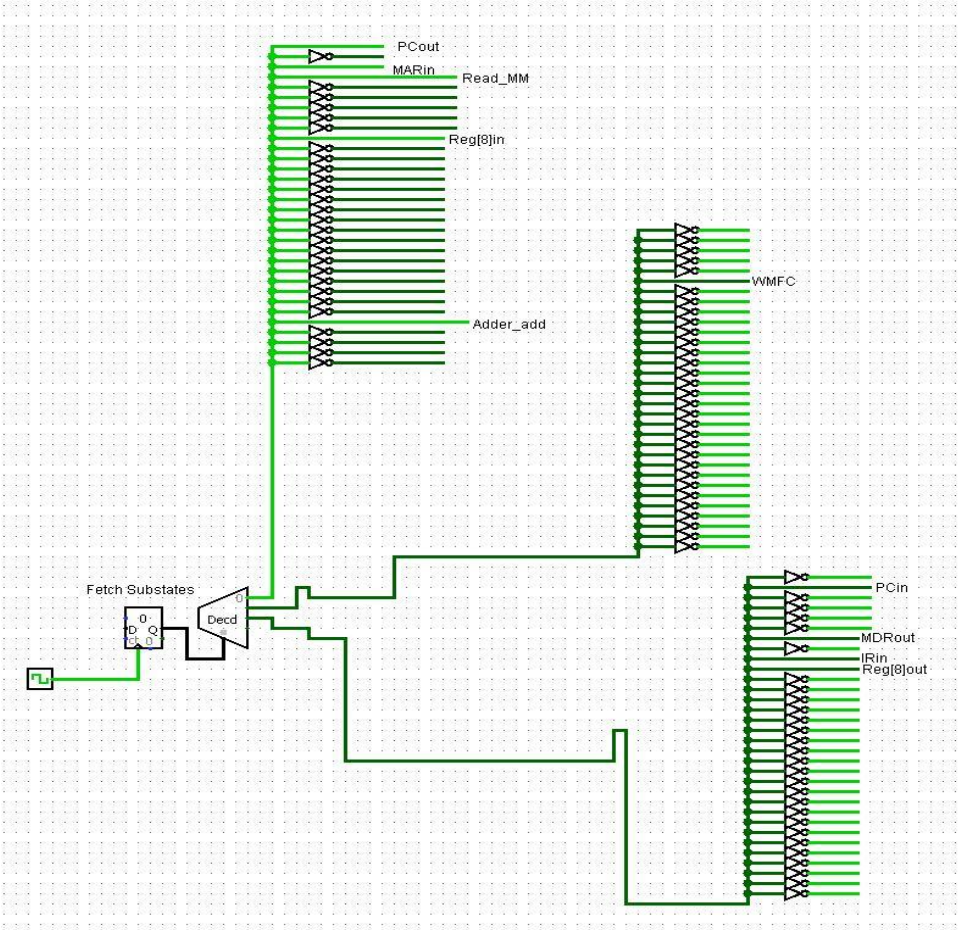

Figure 34: Fetch Substate Control Signal Generation Overview

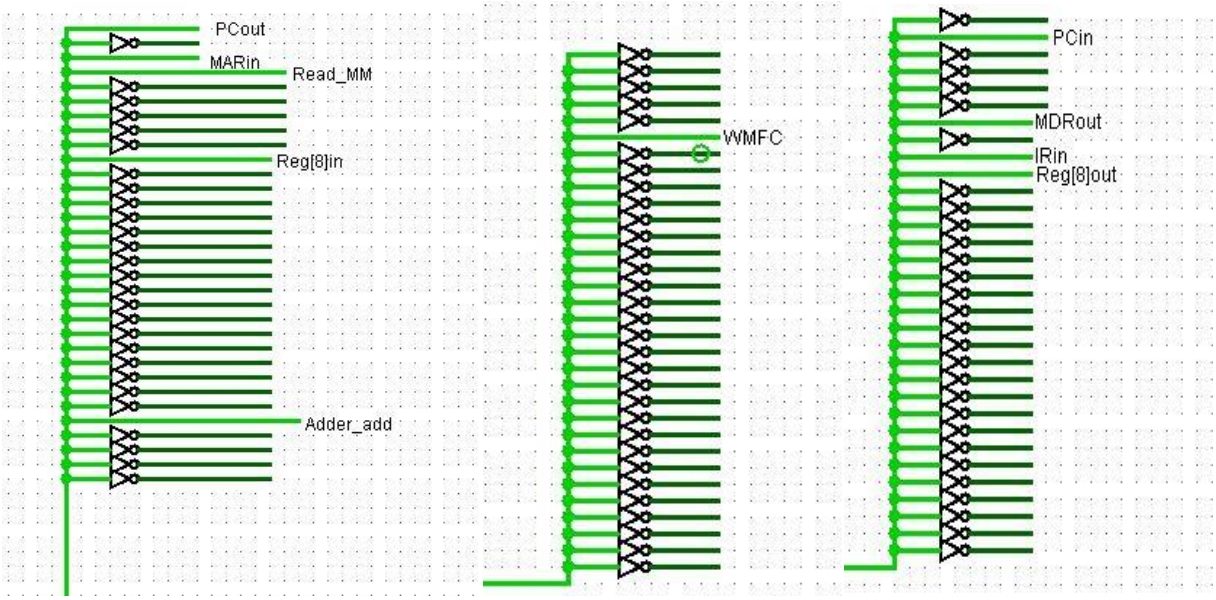

Figure 35: Fetch Substate Step Iteration

VI. Optimizations

The main optimizations made to this machine are as follows:

1. Implementing the ALU with a carry lookahead adder
2. Implementing a separate ALU control unit
3. Adding extra temporary registers (OP1 and OP2) to the data path
4. Adding an autoincrement to the program counter via an adder and additional register (Reg8) in the register file
5. Adding the substate sequencer to the finite state machine in the control unit
6. Further optimizing the substates using combinational logic

The tradeoffs and justifications for each optimization will be discussed in this section.

1. Carry Lookahead Adder

Tradeoffs

Negatives: The carry lookahead adder uses more hardware than a ripple carry adder design, and is more complex to design.

Positives: The carry lookahead adder is faster because it calculates the carry bit(s) of the arithmetic operation before computing the sum of the operation

Justification

The reduced gate delay for adding operations outweighs the extra hardware and design time required for the CLA, because 9 of the 16 instructions require adding. This has a sizeable impact on operation time in this machine.

2. ALU Control Unit

Tradeoffs

Negatives: Implementing the ALU control unit requires extra hardware and design time

Positives: Reduces the size of the main control unit, and reduces the number of control signals by 6 through use of a decoder and combinational logic based on opcode substates.

Justification

While implementing the separate ALU control unit does require extra hardware, it makes the implementation of the main control unit less complex by reducing the number of control signals. In this machine, not implementing the ALU control unit would have added 6 more control signals, causing the unused control signal output bits to increase from 2 to 32 due to requiring a 64 bit output.

3. Adding Extra Temporary Registers (OP1 and OP2) to the Data Path

Tradeoffs

Negatives: Requires adding 2 additional registers and 4 extra control signals

Positives: Makes the operations involving OP1 and OP2 easier to implement, and reduces combinational logic in the control unit

Justification

Adding OP1 and OP2 to the datapath is justified because these operands are used frequently in the machine instructions. Adding them requires a minimal hardware increase, but reduces the control unit delay. This is because the register file does not have to be addressed every single time OP1 and OP2 need to be used.

4. Autoincrementing the Program Counter

Tradeoffs

Negatives: Requires adding an additional register and adder to the register file

Positives: Reduces instruction fetch time by several clock cycles due to not having to access the ALU for every PC increment

Justification

The cost of adding the extra hardware is warranted because of the frequent nature of the fetch cycle. If the adder and Reg8 were not implemented the machine would be much slower due to the 3 extra clock cycles added to each fetch cycle to increment the PC with the ALU.

5. Adding the Substate Sequencer to the Finite State Machine in the Control Unit

Tradeoffs

Negatives: Requires additional design time and hardware

Positives: Requires overall less hardware than implementing a FSM with 50+ states, is much more feasible to design and implement

Justification

Adding the substate sequencer is the most consequential optimization of this machine. Implementing it reduces the required number flip flops by 33% and the number of states in the state machine by 80%. Actually implementing a FSM with 50+ states is drastically more complex than with the substate sequencer. Without it the implementation of the main part of the control unit shown in the **Appendix** would have been impossible. The substate sequencer is practical because the majority of the substates for each state execute unconditionally, and can therefore be implemented with combinational logic as opposed to sequential logic.

6. Further Substate Optimization using Combinational Logic

Tradeoffs

Negatives: Requires additional design time and hardware

Positives: Reduces the number of substates in the substate sequencer by reusing substates

Justification

Many machine operations share substate instructions. When two different operations share a substate it can be reused. If they differ, combinational logic (and gate) can be used to switch between the substate depending on the opcode. An example of this with the Timeout and Program Check Violation processes is shown in **Figure 36 below.** This reduces the size of the circuit a considerable amount, which justifies the additional design time and hardware. It also reduces control delay slightly, however that would most likely not be noticeable in this machine.

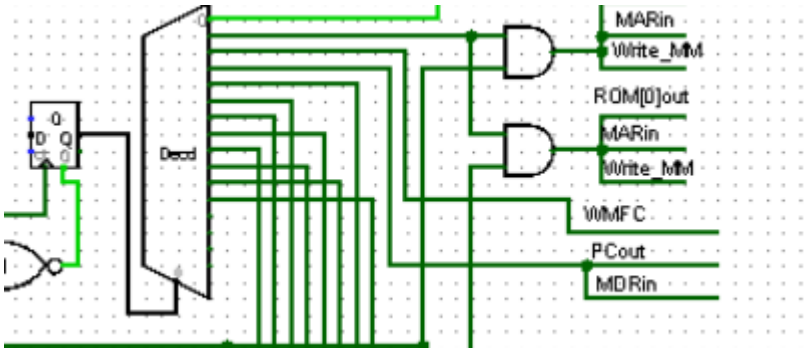

Figure 36: Substate Reduction with Combinational Logic

VII. Appendix

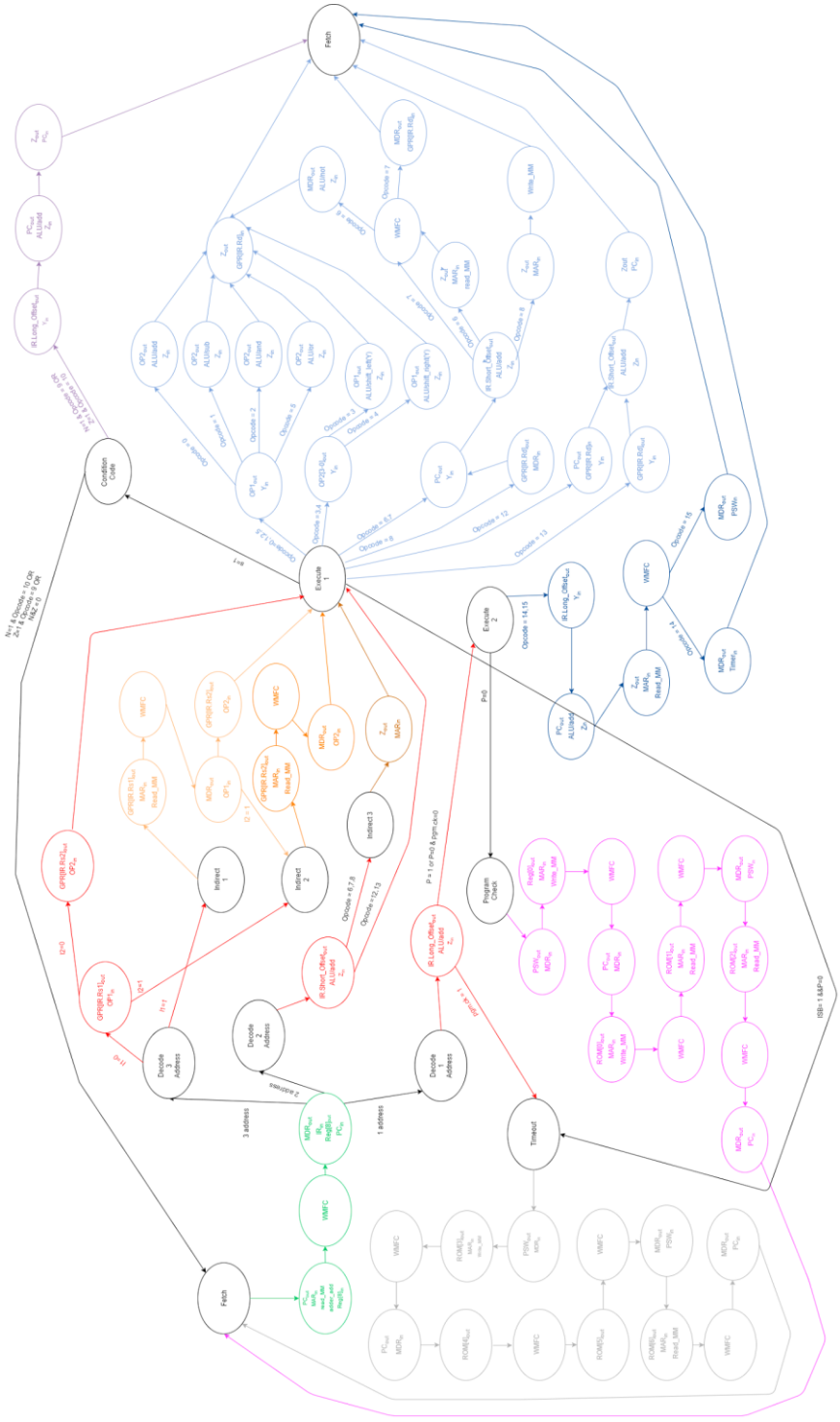

Figure 37: Complete State Diagram including Substates

Current State					Inputs																										Next State				
Q3	Q2	Q1	Q0	sub	3 add	2 add	1 add	I1	I2	Op 0	Op 1	Op 2	Op 3	Op 4	Op 5	Op 6	Op 7	Op 8	Op 9	Op 10	Op 11	Op 12	Op 13	Op 14	Op 15	pgm. ck	S	PSW.P	PSW.N	PSW.Z	ISB	Q3*	Q2*	Q1*	Q0*	sub
0	0	0	0	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
0	0	0	0	(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(3)
0	0	0	0	(3)	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	1	
0	0	0	1		x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	1	(1)
0	0	0	1		x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	
0	0	0	1	(1)	x	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	1	(2)
0	0	0	1	(1)	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	1	
0	0	0	1	(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	
0	0	0	0	(3)	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	1	0	
0	0	1	0		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	1	0	(1)
0	0	1	0	(1)	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	0	
0	0	1	0	(1)	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	0	
0	0	1	0	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	0	
0	0	1	0	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	
0	0	1	0	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	
0	0	0	0	(3)	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	1	1	
0	0	1	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	1	1	(1)
0	0	1	1	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	1	0	0	0	
0	0	1	1	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	x	0	x	x	x	1	0	0	0	
0	0	1	1	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	1	0	1	0	
0	1	0	0		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	(1)
0	1	0	0	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	(2)
0	1	0	0	(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	(3)
0	1	0	0	(3)	x	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	(4)
0	1	0	0	(3)	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	1	
0	1	0	0	(4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	
0	1	0	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	1	(1)
0	1	0	1	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	1	(2)
0	1	0	1	(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	1	(3)
0	1	0	1	(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	
0	1	1	0		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	0	(1)
0	1	1	0	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	
0	1	1	1		x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	add(1)
0	1	1	1		x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	sub(1)
0	1	1	1		x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	and(1)
0	1	1	1		x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	or(1)
0	1	1	1	add(1)	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	add(2)
0	1	1	1	sub(1)	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	sub(2)
0	1	1	1	and(1)	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	and(2)
0	1	1	1	or(1)	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	or(2)
0	1	1	1	add(2)	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	add(3)
0	1	1	1	sub(2)	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	sub(3)
0	1	1	1	and(2)	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	and(3)
0	1	1	1	or(2)	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	or(3)
0	1	1	1	add(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
0	1	1	1	sub(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
0	1	1	1	and(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
0	1	1	1	or(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
0	1	1	1		x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	0	1	1	1	shl(1)
0	1	1	1		x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	0	1	1	1	shra(1)
0	1	1	1	shl(1)	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	0	1	1	1	shl(2)
0	1	1	1	shra(1)	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	0	1	1	1	shra(2)
0	1	1	1	shl(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	0	1	1	1	shl(3)
0	1	1	1	shra(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	0	1	1	1	shra(3)
0	1	1	1	shl(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	0	0	0	0	(1)
0	1	1	1	shra(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	0	0	0	0	(1)
0	1	1	1		x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	0	1	1	1	not(1)
0	1	1	1		x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	1	ld(1)
0	1	1	1	not(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	not(2)
0	1	1	1	ld(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	ld(2)
0	1	1	1	not(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	not(3)
0	1	1	1	ld(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	0	1	1	1	ld(3)
0	1	1	1	not(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	0	1	1	1	not(4)
0	1	1	1	ld(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	0	1	1	1	ld(4)
0	1	1	1	not(4)	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	0	1	1	1	not(5)
0	1	1	1	ld(4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	0	0	0	0	(1)
0	1	1	1	not(5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	0	1	1	1	not(6)
0	1	1	1	not(6)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	0	0	0	0	(1)
0	1	1	1		x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	1	x	x	0	1	1	1	st(1)
0	1	1	1	st(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	1	x	0	1	1	1	st(2)
0	1	1	1	st(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	1	st(3)
0	1	1	1	st(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	1	0	1	1	1	st(4)
0	1	1	1	st(4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	1	0	1	1	1	st(5)
0	1	1	1	st(5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	1	0	0	0	0	(1)
0	1	1	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	1	0	1	1	1	jsr(1)
0	1	1	1	jsr(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	0	1	1	1	jsr(2)
0	1	1	1	jsr(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	0	1	1	1	jsr(3)
0	1	1	1	jsr(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	0	0	0	0	(1)
0	1	1	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	0	1	1	1	rts(1)
0	1	1	1	rts(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	0	1	1	1	rts(2)
0	1	1	1	rts(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	0	1	1	1	rts(3)
0	1	1	1	rts(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	0	0	0	0	(1)
0	1	1	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	0	0	1	

1	0	0	0		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	1	0	0	0	clk(1)
1	0	0	0		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	1	0	0	0	lpsw(1)
1	0	0	0	clk(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	clk(2)
1	0	0	0	lpsw(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	lpsw(2)
1	0	0	0	clk(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	clk(3)
1	0	0	0	lpsw(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	lpsw(3)
1	0	0	0	clk(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	clk(4)
1	0	0	0	lpsw(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	lpsw(4)
1	0	0	0	clk(4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	1	0	0	0	clk(5)
1	0	0	0	lpsw(4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	1	0	0	0	lpsw(5)
1	0	0	0	clk(5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
1	0	0	0	lpsw(5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
1	0	0	0		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	x	x	x	1	0	1	1	
1	0	0	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	1	x	x	1	0	0	1	brn/brz(1)
1	0	0	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	1	x	1	0	0	1	brn/brz(1)
1	0	0	1	brn/brz(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	brn/brz(2)
1	0	0	1	brn/brz(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	brn/brz(2)
1	0	0	1	brn/brz(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	brn/brz(3)
1	0	0	1	brn/brz(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	brn/brz(3)
1	0	0	1	brn/brz(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
1	0	0	1	brn/brz(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
1	0	0	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	1	x	x	0	0	0	0	(1)
1	0	0	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	1	x	0	0	0	0	(1)
1	0	0	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	x	0	0	0	0	(1)
1	0	1	0		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(1)
1	0	1	0	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(2)
1	0	1	0	(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(3)
1	0	1	0	(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(4)
1	0	1	0	(4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(5)
1	0	1	0	(5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(6)
1	0	1	0	(6)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(7)
1	0	1	0	(7)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(8)
1	0	1	0	(8)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(9)
1	0	1	0	(9)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(10)
1	0	1	0	(10)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(11)
1	0	1	0	(11)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	(12)
1	0	1	0	(12)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)
1	0	1	1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(1)
1	0	1	1	(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(2)
1	0	1	1	(2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(3)
1	0	1	1	(3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(4)
1	0	1	1	(4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(5)
1	0	1	1	(5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(6)
1	0	1	1	(6)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(7)
1	0	1	1	(7)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(8)
1	0	1	1	(8)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(9)
1	0	1	1	(9)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(10)
1	0	1	1	(10)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(11)
1	0	1	1	(11)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	1	(12)
1	0	1	1	(12)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	(1)

Table 8: Complete Next State Table including Substates

	Stages/Substate Steps												
Program Check/Trap (1011)	1	2	3	4	5	6	7	8	9	10	11	12	
PCout	0	0	x	1	0	x	0	x	0	0	x	0	
PCin	0	0	x	0	0	x	0	x	0	0	x	0	
MARin	0	1	x	0	1	x	1	x	x	1	x	x	
Read_MM	0	0	x	0		x	1	x	x	1	x	x	
Write_MM	0	1	x	0	1	x	0	x	0	0	x	0	
WMFC	0	0	1	0	0	1	0	1	0	0	1	0	
MDRout	0	0	x	0	0	x	0	x	0	0	x	1	
MDRin	1	0	x	1	0	x	0	x	1	0	x	0	
IRin	x	x	x	x	x	x	x	x	x	x	x	x	
Reg[8]out	x	x	x	x	x	x	x	x	x	x	x	x	
Reg[8]in	x	x	x	x	x	x	x	x	x	x	x	x	
OP1out	x	x	x	x	x	x	x	x	x	x	x	x	
OP1in	x	x	x	x	x	x	x	x	x	x	x	x	
OP2out	x	x	x	x	x	x	x	x	x	x	x	x	
OP2in	x	x	x	x	x	x	x	x	x	x	x	x	
Yin	x	x	x	x	x	x	x	x	x	x	x	x	
Zin	x	x	x	x	x	x	x	x	x	x	x	x	
Zout	x	x	x	x	x	x	x	x	x	x	x	x	
GPRin	x	x	x	x	x	x	x	x	x	x	x	x	
GPRout	x	x	x	x	x	x	x	x	x	x	x	x	
OP2[3-0]out	x	x	x	x	x	x	x	x	x	x	x	x	
IR.Short_Offsetout	x	x	x	x	x	x	x	x	x	x	x	x	
IR.Long_Offsetout	x	x	x	x	x	x	x	x	x	x	x	x	
PSWout	1	0	x	0	0	x	0	x	0	0	x	0	
ROM[out]	0	1	x	0	1	x	1	x	0	1	x	0	
ROM[in]	x	x	x	x	x	x	x	x	x	x	x	x	
PSWin	0	0	x	0	0	x	0	x	1	0	x	0	
adder_add	x	x	x	x	x	x	x	x	x	x	x	x	
ALUOP	x	x	x	x	x	x	x	x	x	x	x	x	
Timerin/CLKin	x	x	x	x	x	x	x	x	x	x	x	x	

A

	Stages/Substate Steps												
ADD (OPCODE 0)	1	2	3	4	5	6	7	8	9	10	11	12	
PCout	x	x	x										
PCin	x	x	x										
MARin	x	x	x										
Read_MM	x	x	x										
Write_MM	x	x	x										
WMFC	x	x	x										
MDRout	x	x	x										
MDRin	x	x	x										
IRin	x	x	x										
Reg[8]out	x	x	x										
Reg[8]in	x	x	x										
OP1out	1	0	0										
OP1in	x	x	x										
OP2out	0	1	0										
OP2in	x	x	x										
Yin	1	0	0										
Zin	0	1	0										
Zout	0	0	1										
GPRin	x	x	x										
GPRout	x	x	x										
OP2[3-0]out	x	x	x										
IR.Short_Offsetout	x	x	x										
IR.Long_Offsetout	x	x	x										
PSWout	x	x	x										
ROM[out]	x	x	x										
ROM[in]	x	x	x										
PSWin	x	x	x										
adder_add	x	x	x										
ALUOP	0	1	0										
Timerin/CLKin	x	x	x										

B

	Stages/Substate Steps												
SUBTRACT (OPCODE 1)	1	2	3	4	5	6	7	8	9	10	11	12	
PCout	x	x	x										
PCin	x	x	x										
MARin	x	x	x										
Read_MM	x	x	x										
Write_MM	x	x	x										
WMFC	x	x	x										
MDRout	x	x	x										
MDRin	x	x	x										
IRin	x	x	x										
Reg[8]out	x	x	x										
Reg[8]in	x	x	x										
OP1out	1	0	0										
OP1in	x	x	x										
OP2out	0	1	0										
OP2in	x	x	x										
Yin	1	0	0										
Zin	0	1	0										
Zout	0	0	1										
GPRin	x	x	x										
GPRout	x	x	x										
OP2[3-0]out	x	x	x										
IR.Short_Offsetout	x	x	x										
IR.Long_Offsetout	x	x	x										
PSWout	x	x	x										
ROM[out]	x	x	x										
ROM[in]	x	x	x										
PSWin	x	x	x										
adder_add	x	x	x										
ALUOP	0	1	0										
Timerin/CLKin	x	x	x										

C

	Stages/Substate Steps												
AND (OPCODE 2)	1	2	3	4	5	6	7	8	9	10	11	12	
PCout	x	x	x										
PCin	x	x	x										
MARin	x	x	x										
Read_MM	x	x	x										
Write_MM	x	x	x										
WMFC	x	x	x										
MDRout	x	x	x										
MDRin	x	x	x										
IRin	x	x	x										
Reg[8]out	x	x	x										
Reg[8]in	x	x	x										
OP1out	1	0	0										
OP1in	x	x	x										
OP2out	0	1	0										
OP2in	x	x	x										
Yin	1	0	0										
Zin	0	1	0										
Zout	0	0	1										
GPRin	x	x	x										
GPRout	x	x	x										
OP2[3-0]out	x	x	x										
IR.Short_Offsetout	x	x	x										
IR.Long_Offsetout	x	x	x										
PSWout	x	x	x										
ROM[out]	x	x	x										
ROM[in]	x	x	x										
PSWin	x	x	x										
adder_add	x	x	x										
ALUOP	0	1	0										
Timerin/CLKin	x	x	x										

D

	Stages/Substate Steps											
SHL (OPCODE 3)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x	x	x									
PCin	x	x	x									
MARin	x	x	x									
Read_MM	x	x	x									
Write_MM	x	x	x									
WMFC	x	x	x									
MDRout	x	x	x									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	x	x	x									
OP1in	x	x	x									
OP2out	0	1	0									
OP2in	x	x	x									
Yin	1	0	0									
Zin	0	1	0									
Zout	0	0	1									
GPRin	0	0	1									
GPRout	x	x	x									
OP2[3-0]out	1	0	0									
IR.Short_Offsetout	x	x	x									
IR.Long_Offsetout	x	x	x									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALU/OP	0	1	0									
Timerin/CLKin	x	x	x									

E

	Stages/Substate Steps											
SHRA (OPCODE 4)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x	x	x									
PCin	x	x	x									
MARin	x	x	x									
Read_MM	x	x	x									
Write_MM	x	x	x									
WMFC	x	x	x									
MDRout	x	x	x									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	x	x	x									
OP1in	x	x	x									
OP2out	0	1	0									
OP2in	x	x	x									
Yin	1	0	0									
Zin	0	1	0									
Zout	0	0	1									
GPRin	0	0	1									
GPRout	x	x	x									
OP2[3-0]out	1	0	0									
IR.Short_Offsetout	x	x	x									
IR.Long_Offsetout	x	x	x									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALU/OP	0	1	0									
Timerin/CLKin	x	x	x									

F

	Stages/Substate Steps											
OR (OPCODE 5)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x	x	x									
PCin	x	x	x									
MARin	x	x	x									
Read_MM	x	x	x									
Write_MM	x	x	x									
WMFC	x	x	x									
MDRout	x	x	x									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	1	0	0									
OP1in	x	x	x									
OP2out	0	1	0									
OP2in	x	x	x									
Yin	1	0	0									
Zin	0	1	0									
Zout	0	0	1									
GPRin	0	0	1									
GPRout	x	x	x									
OP2[3-0]out	x	x	x									
IR.Short_Offsetout	x	x	x									
IR.Long_Offsetout	x	x	x									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALU/OP	0	1	0									
Timerin/CLKin	x	x	x									

G

	Stages/Substate Steps											
NOT (OPCODE 6)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	1	0	0	0	0	0						
PCin	x	x	x	x	x	x						
MARin	0	0	1	0	0	0						
Read_MM	0	0	1	0	0	0						
Write_MM	x	x	x	x	x	x						
WMFC	0	0	0	1	0	0						
MDRout	0	0	0	0	1	0						
MDRin	x	x	x	x	x	x						
IRin	x	x	x	x	x	x						
Reg[8]out	x	x	x	x	x	x						
Reg[8]in	x	x	x	x	x	x						
OP1out	x	x	x	x	x	x						
OP1in	x	x	x	x	x	x						
OP2out	x	x	x	x	x	x						
OP2in	x	x	x	x	x	x						
Yin	1	0	0	0	0	0						
Zin	0	1	0	0	1	0						
Zout	0	0	1	0	0	1						
GPRin	0	0	0	0	0	1						
GPRout	x	x	x	x	x	x						
OP2[3-0]out	x	x	x	x	x	x						
IR.Short_Offsetout	0	1	0	0	0	0						
IR.Long_Offsetout	x	x	x	x	x	x						
PSWout	x	x	x	x	x	x						
ROM[out]	x	x	x	x	x	x						
ROM[in]	x	x	x	x	x	x						
PSWin	x	x	x	x	x	x						
adder_add	x	x	x	x	x	x						
ALU/OP	0	1	0	0	1	0						
Timerin/CLKin	x	x	x	x	x	x						

H

	Stages/Substate Steps											
LOAD (OPCODE 7)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	1	0	0	0	0							
PCin	x	x	x	x	x							
MARin	0	0	1	0	0							
Read_MM	0	0	1	0	0							
Write_MM	x	x	x	x	x							
WMFC	0	0	0	1	0							
MDRout	0	0	0	0	1							
MDRin	x	x	x	x	x							
IRin	x	x	x	x	x							
Reg[8]out	x	x	x	x	x							
Reg[8]in	x	x	x	x	x							
OP1out	x	x	x	x	x							
OP1in	x	x	x	x	x							
OP2out	x	x	x	x	x							
OP2in	x	x	x	x	x							
Yin	1	0	0	0	0							
Zin	0	1	0	0	0							
Zout	0	0	1	0	0							
GPRin	0	0	0	0	1							
GPRout	x	x	x	x	x							
OP2[3-0]out	x	x	x	x	x							
IR.Short_Offsetout	0	1	0	0	0							
IR.Long_Offsetout	x	x	x	x	x							
PSWout	x	x	x	x	x							
ROM[out]	x	x	x	x	x							
ROM[in]	x	x	x	x	x							
PSWin	x	x	x	x	x							
adder_add	x	x	x	x	x							
ALU/OP	0	1	0	0	0							
Timerin/CLKin	x	x	x	x	x							

I

	Stages/Substate Steps											
STORE (OPCODE 8)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	0	1	0	0	0							
PCin	x	x	x	x	x							
MARin	0	0	0	1	0							
Read_MM	x	x	x	x	x							
Write_MM	0	0	0	0	1							
WMFC	x	x	x	x	x							
MDRout	x	x	x	x	x							
MDRin	1	0	0	0	0							
IRin	x	x	x	x	x							
Reg[8]out	x	x	x	x	x							
Reg[8]in	x	x	x	x	x							
OP1out	x	x	x	x	x							
OP1in	x	x	x	x	x							
OP2out	x	x	x	x	x							
OP2in	x	x	x	x	x							
Yin	0	1	0	0	0							
Zin	0	0	1	0	0							
Zout	0	0	0	1	0							
GPRin	x	x	x	x	x							
GPRout	1	0	0	0	0							
OP2[3-0]out	x	x	x	x	x							
IR.Short_Offsetout	0	0	1	0	0							
IR.Long_Offsetout	x	x	x	x	x							
PSWout	x	x	x	x	x							
ROM[out]	x	x	x	x	x							
ROM[in]	x	x	x	x	x							
PSWin	x	x	x	x	x							
adder_add	x	x	x	x	x							
ALU/OP	0	0	1	0	0							
Timerin/CLKin	x	x	x	x	x							

J

	Stages/Substate Steps											
Branch Negative (OPCODE9)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	0	1	0									
PCin	0	0	1									
MARin	x	x	x									
Read_MM	x	x	x									
Write_MM	x	x	x									
WMFC	x	x	x									
MDRout	x	x	x									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	x	x	x									
OP1in	x	x	x									
OP2out	x	x	x									
OP2in	x	x	x									
Yin	1	0	0									
Zin	0	1	1									
Zout	x	x	x									
GPRin	x	x	x									
GPRout	x	x	x									
OP2[3-0]out	x	x	x									
IR.Short_Offsetout	x	x	x									
IR.Long_Offsetout	1	0	0									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALU/OP	0	1	0									
Timerin/CLKin	x	x	x									

K

	Stages/Substate Steps											
Branch Zero (OPCODE 10)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	0	1	0									
PCin	0	0	1									
MARin	x	x	x									
Read_MM	x	x	x									
Write_MM	x	x	x									
WMFC	x	x	x									
MDRout	x	x	x									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	x	x	x									
OP1in	x	x	x									
OP2out	x	x	x									
OP2in	x	x	x									
Yin	1	0	0									
Zin	0	1	1									
Zout	x	x	x									
GPRin	x	x	x									
GPRout	x	x	x									
OP2[3-0]out	x	x	x									
IR.Short_Offsetout	x	x	x									
IR.Long_Offsetout	1	0	0									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALU/OP	0	1	0									
Timerin/CLKin	x	x	x									

L

Branch (OPCODE 11)	Stages/Substate Steps											
	1	2	3	4	5	6	7	8	9	10	11	12
PCout	0	1	1									
PCin	x	x	x									
MARin	x	x	x									
Read_MM	x	x	x									
Write_MM	x	x	x									
WMFC	x	x	x									
MDRout	x	x	x									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	x	x	x									
OP1in	x	x	x									
OP2out	x	x	x									
OP2in	x	x	x									
Yin	1	0	0									
Zin	0	1	0									
Zout	0	0	1									
GPRin	x	x	x									
GPRout	x	x	x									
OP2[3-0]out	x	x	x									
IR.Short_Offsetout	x	x	x									
IR.Long_Offsetout	1	0	0									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALU/OP	0	1	0									
Timerin/CLKin	x	x	x									

M

JSR (OPCODE 12)	Stages/Substate Steps											
	1	2	3	4	5	6	7	8	9	10	11	12
PCout	1	0	0									
PCin	0	0	1									
MARin	x	x	x									
Read_MM	x	x	x									
Write_MM	x	x	x									
WMFC	x	x	x									
MDRout	x	x	x									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	x	x	x									
OP1in	x	x	x									
OP2out	x	x	x									
OP2in	x	x	x									
Yin	1	0	0									
Zin	0	1	0									
Zout	0	0	1									
GPRin	1	0	0									
GPRout	x	x	x									
OP2[3-0]out	x	x	x									
IR.Short_Offsetout	0	1	0									
IR.Long_Offsetout	x	x	x									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALU/OP	0	1	0									
Timerin/CLKin	x	x	x									

N

RTS (OPCODE 13)	Stages/Substate Steps											
	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x	x	x									
PCin	0	0	1									
MARin	x	x	x									
Read_MM	x	x	x									
Write_MM	x	x	x									
WMFC	x	x	x									
MDRout	x	x	x									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	x	x	x									
OP1in	x	x	x									
OP2out	x	x	x									
OP2in	x	x	x									
Yin	1	0	0									
Zin	0	1	0									
Zout	0	0	1									
GPRin	x	x	x									
GPRout	1	0	0									
OP2[3-0]out	x	x	x									
IR.Short_Offsetout	0	1	0									
IR.Long_Offsetout	x	x	x									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALU/OP	0	1	0									
Timerin/CLKin	x	x	x									

O

CLK (OPCODE 14)	Stages/Substate Steps											
	1	2	3	4	5	6	7	8	9	10	11	12
PCout	0	1	0	x	0							
PCin	x	x	x	x	x							
MARin	0	0	1	x	0							
Read_MM	0	0	1	x	0							
Write_MM	x	x	x	x	x							
WMFC	0	0	0	1	0							
MDRout	0	0	0	x	1							
MDRin	x	x	x	x	x							
IRin	x	x	x	x	x							
Reg[8]out	x	x	x	x	x							
Reg[8]in	x	x	x	x	x							
OP1out	x	x	x	x	x							
OP1in	x	x	x	x	x							
OP2out	x	x	x	x	x							
OP2in	x	x	x	x	x							
Yin	1	0	0	x	0							
Zin	0	1	0	x	0							
Zout	0	0	1	x	0							
GPRin	x	x	x	x	x							
GPRout	x	x	x	x	x							
OP2[3-0]out	x	x	x	x	x							
IR.Short_Offsetout	1	0	0	x	0							
IR.Long_Offsetout	x	x	x	x	x							
PSWout	x	x	x	x	x							
ROM[out]	x	x	x	x	x							
ROM[in]	x	x	x	x	x							
PSWin	x	x	x	x	x							
adder_add	x	x	x	x	x							
ALU/OP	0	1	0	x	0							
Timerin/CLKin	0	0	0	0	1							

P

	Stages/Substate Steps											
LPSW (OPCODE 15)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	0	1	0	x	0							
PCin	x	x	x	x	x							
MARin	0	0	1	x	0							
Read_MM	0	0	1	x	0							
Write_MM	x	x	x	x	x							
WMFC	0	0	0	1	0							
MDRout	0	0	0	x	1							
MDRin	x	x	x	x	x							
IRin	x	x	x	x	x							
Reg[8]out	x	x	x	x	x							
Reg[8]in	x	x	x	x	x							
OP1out	x	x	x	x	x							
OP1in	x	x	x	x	x							
OP2out	x	x	x	x	x							
OP2in	x	x	x	x	x							
Yin	1	0	0	x	0							
Zin	0	1	0	x	0							
Zout	0	0	1	x	0							
GPRin	x	x	x	x	x							
GPRout	x	x	x	x	x							
OP2[3-0]out	x	x	x	x	x							
IR.Short_Offsetout	1	0	0	x	0							
IR.Long_Offsetout	x	x	x	x	x							
PSWout	x	x	x	x	x							
ROM[out]	x	x	x	x	x							
ROM[in]	x	x	x	x	x							
PSWin	0	0	0	0	1							
adder_add	x	x	x	x	x							
ALUOP	0	1	0	x	0							
Timerin/CLKin	0	0	0	0	1							

Q

	Stages/Substate Steps											
DECODE3ADDR when I=0 (0001)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x	x										
PCin	x	x										
MARin	x	x										
Read_MM	x	x										
Write_MM	x	x										
WMFC	x	x										
MDRout	x	x										
MDRin	x	x										
IRin	x	x										
Reg[8]out	x	x										
Reg[8]in	x	x										
OP1out	x	x										
OP1in	1	0										
OP2out	x	x										
OP2in	0	1										
Yin	x	x										
Zin	x	x										
Zout	x	x										
GPRin	x	x										
GPRout	1	1										
OP2[3-0]out	x	x										
IR.Short_Offsetout	x	x										
IR.Long_Offsetout	x	x										
PSWout	x	x										
ROM[out]	x	x										
ROM[in]	x	x										
PSWin	x	x										
adder_add	x	x										
ALUOP	x	x										
Timerin/CLKin	x	x										

R

	Stages/Substate Steps											
DECODE2ADDR (0010)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x											
PCin	x											
MARin	x											
Read_MM	x											
Write_MM	x											
WMFC	x											
MDRout	x											
MDRin	x											
IRin	x											
Reg[8]out	x											
Reg[8]in	x											
OP1out	x											
OP1in	x											
OP2out	x											
OP2in	x											
Yin	x											
Zin	1											
Zout	x											
GPRin	x											
GPRout	x											
OP2[3-0]out	x											
IR.Short_Offsetout	1											
IR.Long_Offsetout	x											
PSWout	x											
ROM[out]	x											
ROM[in]	x											
PSWin	x											
adder_add	x											
ALUOP	1											
Timerin/CLKin	x											

S

	Stages/Substate Steps											
DECODE1ADDR (0011)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x											
PCin	x											
MARin	x											
Read_MM	x											
Write_MM	x											
WMFC	x											
MDRout	x											
MDRin	x											
IRin	x											
Reg[8]out	x											
Reg[8]in	x											
OP1out	x											
OP1in	x											
OP2out	x											
OP2in	x											
Yin	x											
Zin	1											
Zout	x											
GPRin	x											
GPRout	x											
OP2[3-0]out	x											
IR.Short_Offsetout	x											
IR.Long_Offsetout	1											
PSWout	x											
ROM[out]	x											
ROM[in]	x											
PSWin	x											
adder_add	x											
ALUOP	1											
Timerin/CLKin	x											

T

	Stages/Substate Steps											
INDIRECT 3 (0110)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x											
PCin	x											
MARin	1											
Read_MM	x											
Write_MM	x											
WMFC	x											
MDRout	x											
MDRin	x											
IRin	x											
Reg[8]out	x											
Reg[8]in	x											
OP1out	x											
OP1in	x											
OP2out	x											
OP2in	x											
Yin	x											
Zin	x											
Zout	1											
GPRin	x											
GPRout	x											
OP2[3-0]out	x											
IR.Short_Offsetout	x											
IR.Long_Offsetout	x											
PSWout	x											
ROM[out]	x											
ROM[in]	x											
PSWin	x											
adder_add	x											
ALUOP	x											
Timerin/CLKin	x											

U

	Stages/Substate Steps											
Timeout (1010)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	0	0	x	1	0	x	0	x	0	0	x	0
PCin	0	0	x	0	0	x	0	x	0	0	x	1
MARin	0	1	x	0	0	x	0	x	0	1	x	0
Read_MM	0	0	x	0	0	x	0	x	0	1	x	0
Write_MM	0	1	x	0	0	x	0	x	0	0	x	0
WMFC	0	0	1	0	0	1	0	1	0	0	1	0
MDRout	0	0	x	0	0	x	0	x	1	0	x	1
MDRin	1	0	x	1	0	x	0	x	0	0	x	0
IRin	x	x	x	x	x	x	x	x	x	x	x	x
Reg[8]out	x	x	x	x	x	x	x	x	x	x	x	x
Reg[8]in	x	x	x	x	x	x	x	x	x	x	x	x
OP1out	x	x	x	x	x	x	x	x	x	x	x	x
OP1in	x	x	x	x	x	x	x	x	x	x	x	x
OP2out	x	x	x	x	x	x	x	x	x	x	x	x
OP2in	x	x	x	x	x	x	x	x	x	x	x	x
Yin	x	x	x	x	x	x	x	x	x	x	x	x
Zin	x	x	x	x	x	x	x	x	x	x	x	x
Zout	x	x	x	x	x	x	x	x	x	x	x	x
GPRin	x	x	x	x	x	x	x	x	x	x	x	x
GPRout	x	x	x	x	x	x	x	x	x	x	x	x
OP2[3-0]out	x	x	x	x	x	x	x	x	x	x	x	x
IR.Short_Offsetout	x	x	x	x	x	x	x	x	x	x	x	x
IR.Long_Offsetout	x	x	x	x	x	x	x	x	x	x	x	x
PSWout	1	0	x	0	0	x	0	x	0	0	x	x
ROM[out]	0	1	x	0	1	x	1	x	0	1	x	x
ROM[in]	x	x	x	x	x	x	x	x	x	x	x	x
PSWin	0	0	x	0	0	x	0	x	1	0	x	x
adder_add	x	x	x	x	x	x	x	x	x	x	x	x
ALUOP	x	x	x	x	x	x	x	x	x	x	x	x
Timerin/CLKin	x	x	x	x	x	x	x	x	x	x	x	x

V

	Stages/Substate Steps											
INDIRECT 1 (0100)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x	x	x	x								
PCin	x	x	x	x								
MARin	1	x	0	0								
Read_MM	1	x	0	0								
Write_MM	x	x	x	x								
WMFC	0	1	0	0								
MDRout	0	x	1	0								
MDRin	x	x	x	x								
IRin	x	x	x	x								
Reg[8]out	x	x	x	x								
Reg[8]in	x	x	x	x								
OP1out	x	x	x	x								
OP1in	0	x	1	0								
OP2out	x	x	x	x								
OP2in	0	x	0	1								
Yin	x	x	x	x								
Zin	x	x	x	x								
Zout	x	x	x	x								
GPRin	x	x	x	x								
GPRout	1	x	0	1								
OP2[3-0]out	x	x	x	x								
IR.Short_Offsetout	x	x	x	x								
IR.Long_Offsetout	x	x	x	x								
PSWout	x	x	x	x								
ROM[out]	x	x	x	x								
ROM[in]	x	x	x	x								
PSWin	x	x	x	x								
adder_add	x	x	x	x								
ALUOP	x	x	x	x								
Timerin/CLKin	x	x	x	x								

W

	Stages/Substate Steps											
INDIRECT 2 (0101)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x	x	x									
PCin	x	x	x									
MARin	1	x	0									
Read_MM	1	x	0									
Write_MM	x	x	x									
WMFC	0	1	0									
MDRout	0	x	1									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	x	x	x									
OP1in	0	x	1									
OP2out	x	x	x									
OP2in	0	x	0									
Yin	x	x	x									
Zin	x	x	x									
Zout	x	x	x									
GPRin	x	x	x									
GPRout	1	x	0									
OP2[3-0]out	x	x	x									
IR.Short_Offsetout	x	x	x									
IR.Long_Offsetout	x	x	x									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALUOP	x	x	x									
Timerin/CLKin	x	x	x									

X

	Stages/Substate Steps											
INDIRECT 1 (0100)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x	x	x	x								
PCin	x	x	x	x								
MARin	1	x	0	0								
Read_MM	1	x	0	0								
Write_MM	x	x	x	x								
WMFC	0	1	0	0								
MDRout	0	x	1	0								
MDRin	x	x	x	x								
IRin	x	x	x	x								
Reg[8]out	x	x	x	x								
Reg[8]in	x	x	x	x								
OP1out	x	x	x	x								
OP1in	0	x	1	0								
OP2out	x	x	x	x								
OP2in	0	x	0	1								
Yin	x	x	x	x								
Zin	x	x	x	x								
Zout	x	x	x	x								
GPRin	x	x	x	x								
GPRout	1	x	0	1								
OP2[3-0]out	x	x	x	x								
IR.Short_Offsetout	x	x	x	x								
IR.Long_Offsetout	x	x	x	x								
PSWout	x	x	x	x								
ROM[out]	x	x	x	x								
ROM[in]	x	x	x	x								
PSWin	x	x	x	x								
adder_add	x	x	x	x								
ALUOP	x	x	x	x								
Timerin/CLKin	x	x	x	x								

Y

	Stages/Substate Steps											
INDIRECT 2 (0101)	1	2	3	4	5	6	7	8	9	10	11	12
PCout	x	x	x									
PCin	x	x	x									
MARin	1	x	0									
Read_MM	1	x	0									
Write_MM	x	x	x									
WMFC	0	1	0									
MDRout	0	x	1									
MDRin	x	x	x									
IRin	x	x	x									
Reg[8]out	x	x	x									
Reg[8]in	x	x	x									
OP1out	x	x	x									
OP1in	0	x	1									
OP2out	x	x	x									
OP2in	0	x	0									
Yin	x	x	x									
Zin	x	x	x									
Zout	x	x	x									
GPRin	x	x	x									
GPRout	1	x	0									
OP2[3-0]out	x	x	x									
IR.Short_Offsetout	x	x	x									
IR.Long_Offsetout	x	x	x									
PSWout	x	x	x									
ROM[out]	x	x	x									
ROM[in]	x	x	x									
PSWin	x	x	x									
adder_add	x	x	x									
ALUOP	x	x	x									
Timerin/CLKin	x	x	x									

Z

Table 9(A-Z): Control Signal Derivation Tables

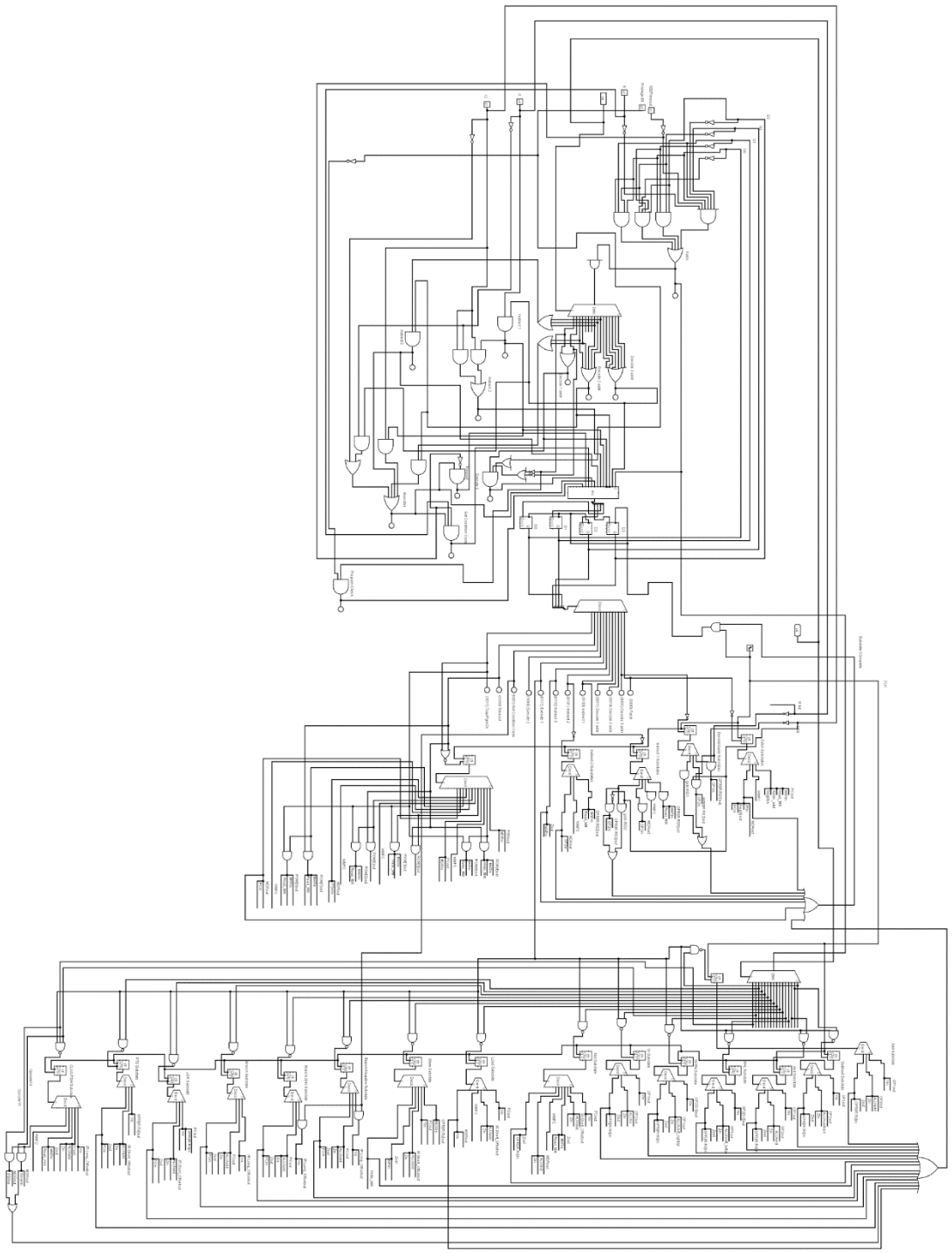

Figure 38: Complete State Machine Implementation (FSM and Substate Sequencer)

VIII. References

"Experimental Computing Laboratory, Philip A. Wilsey." *Experimental Computing Laboratory, Philip A. Wilsey*. University of Cincinnati, n.d. Web. 06 Mar. 2017.

Mano, M. Morris, and Michael D. Ciletti. *Digital Design: With a Introduction to the Verilog Hdl*. Upper Saddle River, NJ: Pearson Prentice Hall, 2013. Print.

Patterson, David A., and John L. Hennessy. *Computer Organization and Design the Hardware/software Interface*. Amsterdam: Morgan Kaufmann, 2016. Print.