Lab 3:: CPU Lab 2
Daquan Smith

CSC 137-02

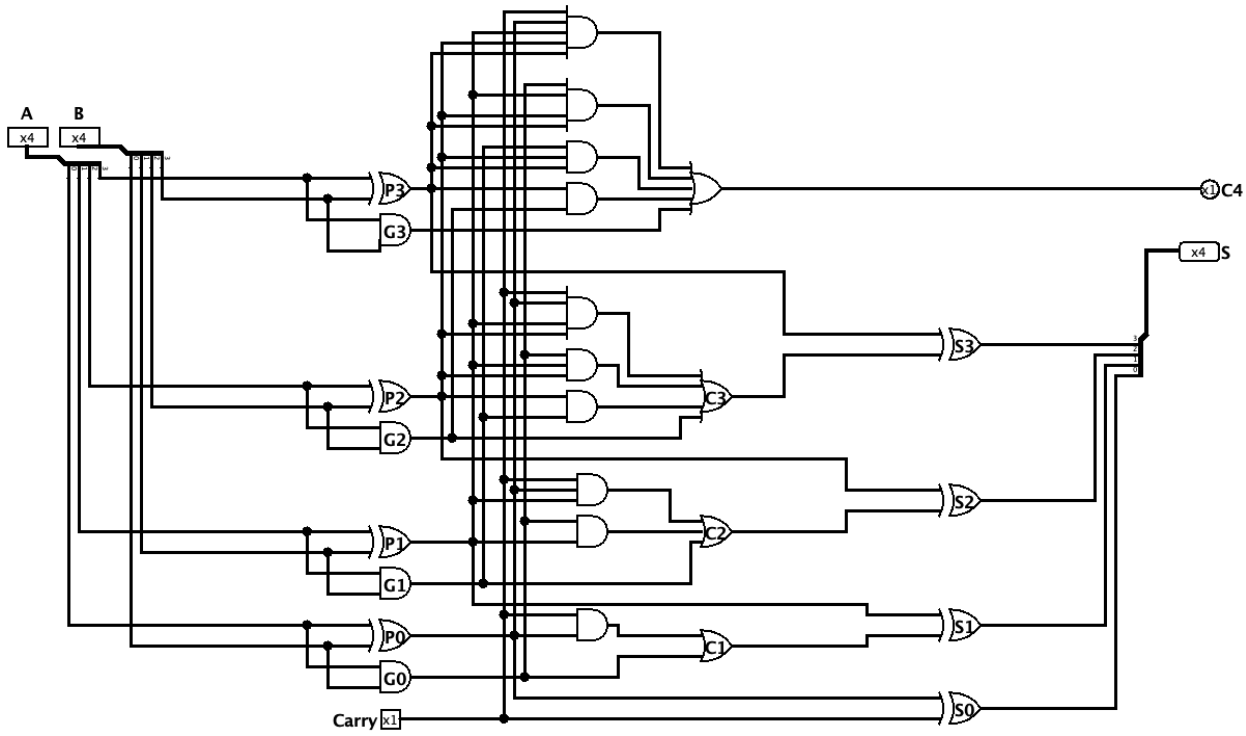
Theory of Operation:

This particular simulator has the task of converting Fahrenheit into Celsius using the RAM of the simulator in order to store the Celsius conversions. This simulator had many more instructions than that of the FISC with additions of branch when carry set, move immediate value, arithmetic shift right, shift left, load, and store. In the first assembly code, we are using the Fahrenheit as an address to access the RAM that stores the 240 or so Celsius conversions. In making the assembler do all of these functions we had to implement a few subcircuits. For starters, we implemented the ALU as its own subcircuit that gathered the information from the instruction line and outputted the correct value to be stored in our register file which is another subcircuit that we made. The way that this register file differs from that of FISCSIM is that the third register does not have the special ability to have its value read and displayed. Instead, we used the addresses of F0 and F1 to store our temperature values. These registers were then read and decoded by our display driver decoder. The ROM (sized 8kX12) that we used contained all of the instructions that were used in order to get the temperatures to display correctly. The simulator itself is not the only thing that extremely differed from FISCSIM. We had a larger bit width in order to accommodate all of the instructions that were used in the MISCAS simulator. This meant that our program counter would store 13 bits instead of the 8 that was from the previous lab. BNZ and BCS used the new addition of the status register in order to know whether or not to jump to the next given address. This meant that we could not simply just add a value to the program counter instead, we would have to add the current address to the target address that was given in these instructions. Finally, another aspect of the simulator that needed to be implemented was our controller which essentially contained all of our enable bits that allowed us to store and load at the correct time as well as write into registers at the correct time. For more details on the aspects of the simulator please see the appendixes A1-A6.

For the second half of the lab, we were tasked with loading the Celsius values into the RAM and displaying the correct Celsius and its corresponding Farenheit temperature in two different ways. In PART A the Celsius was to be stored in the ROM using the corresponding address given by the Fahrenheit. This means when we display the RAM can be used as a translation table to display the correct values. This was made fairly simple using our new move immediate instruction in order to store data. For PART B we are meant to use the actual formula in order to store the values into RAM and display the conversions. This can be done again by using the move immediately to store a constant value of 5/9. Time did not permit me to finish this section of the lab so therefore I will be discussing it and the procedures instead.

APPENDIX 1

FOUR BIT CLA

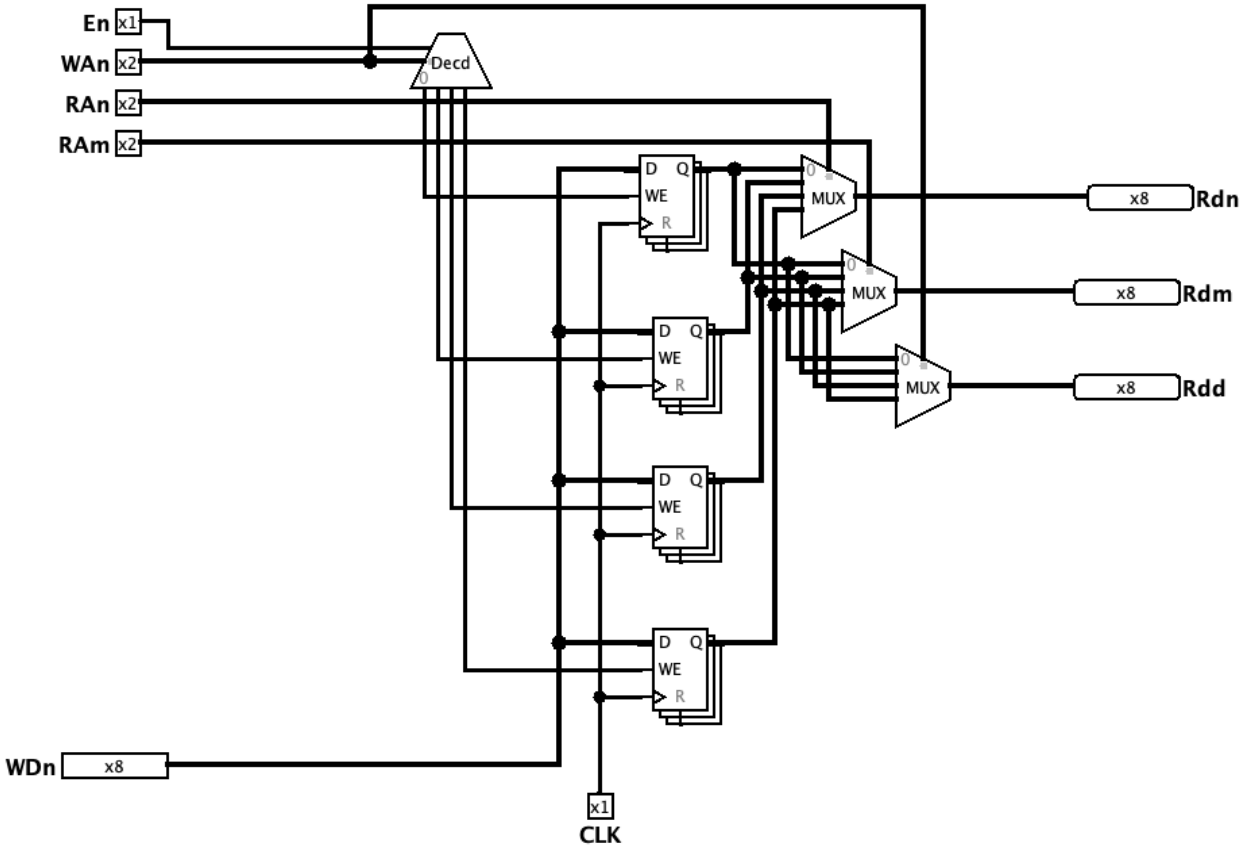


CLA Description:

One of the given restrictions given for this lab is that the use of the given adders in Logisim was only to be used in the program counter calculation. Due to his restraint, there is a need for this four-bit carry look-ahead adder in order for us to increment and decrement values for our subtract and add instructions. In order to build this adder I used the carry propagate formula in order to display carry at the correct time and add each bit. This CLA was then connected by the carry output to another CLA to add the other remaining 4 bits that were in our 8-bit inputs. The output used a splitter to combine these outputs to create our result. The carry flag was set using the second CLA's carry output.

APPENDIX 2

REGISTER FILE

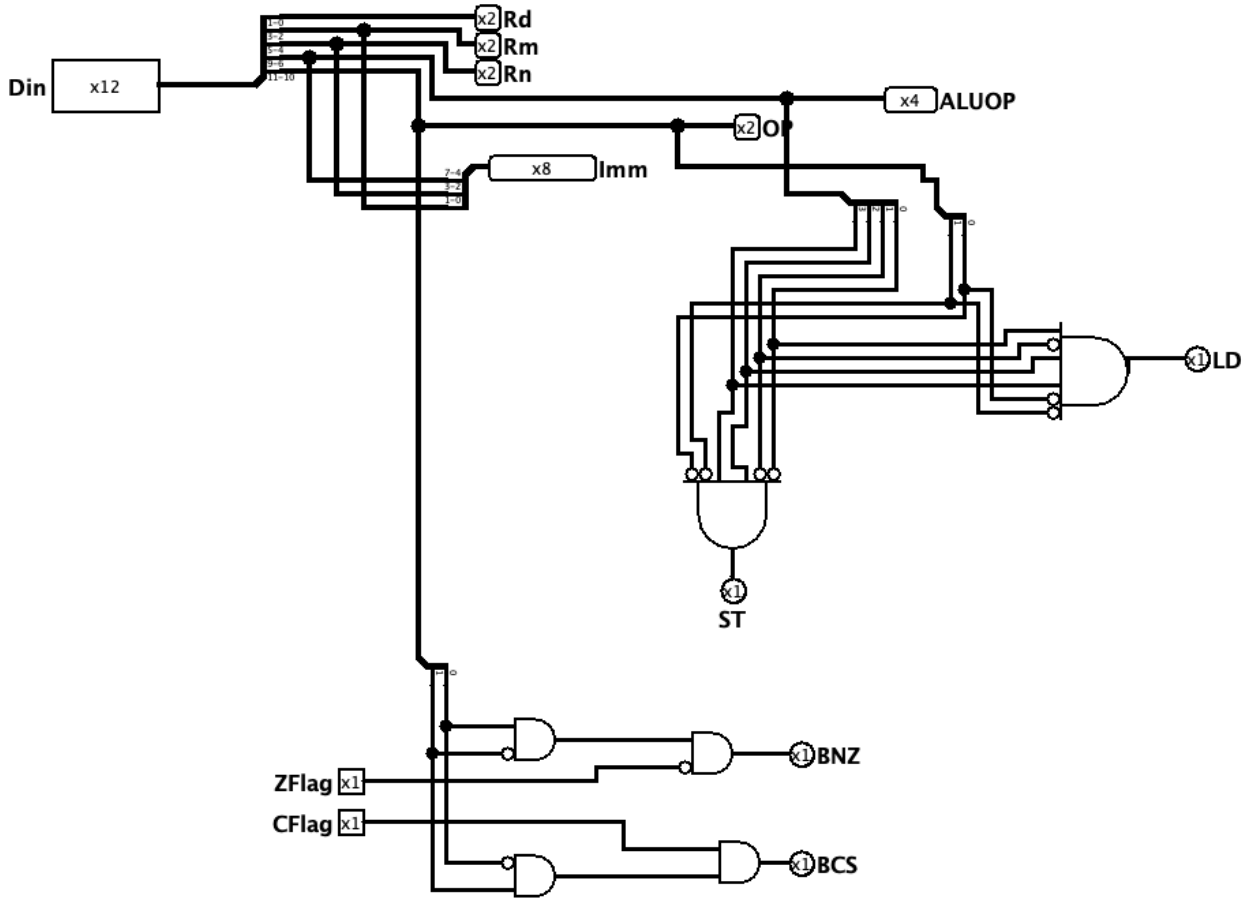


Description:

The register file for this lab is very similar to that of the FISCSIM register file. One thing that is different however is that there is an enabler in the decoder that tells the register file whether or not it is allowed to store. In the main circuit there is the logic that handles this, see Appendix 6. The WAn, RaN, and RAm inputs handle the selection of the Rd, Rn, and Rm registers respectively. These values are fed into the three decoders on the right. The WDn input is the data coming in from our ALU operation or any other operation that is to be done with the values of the registers.

APPENDIX 3

CONTROLLER

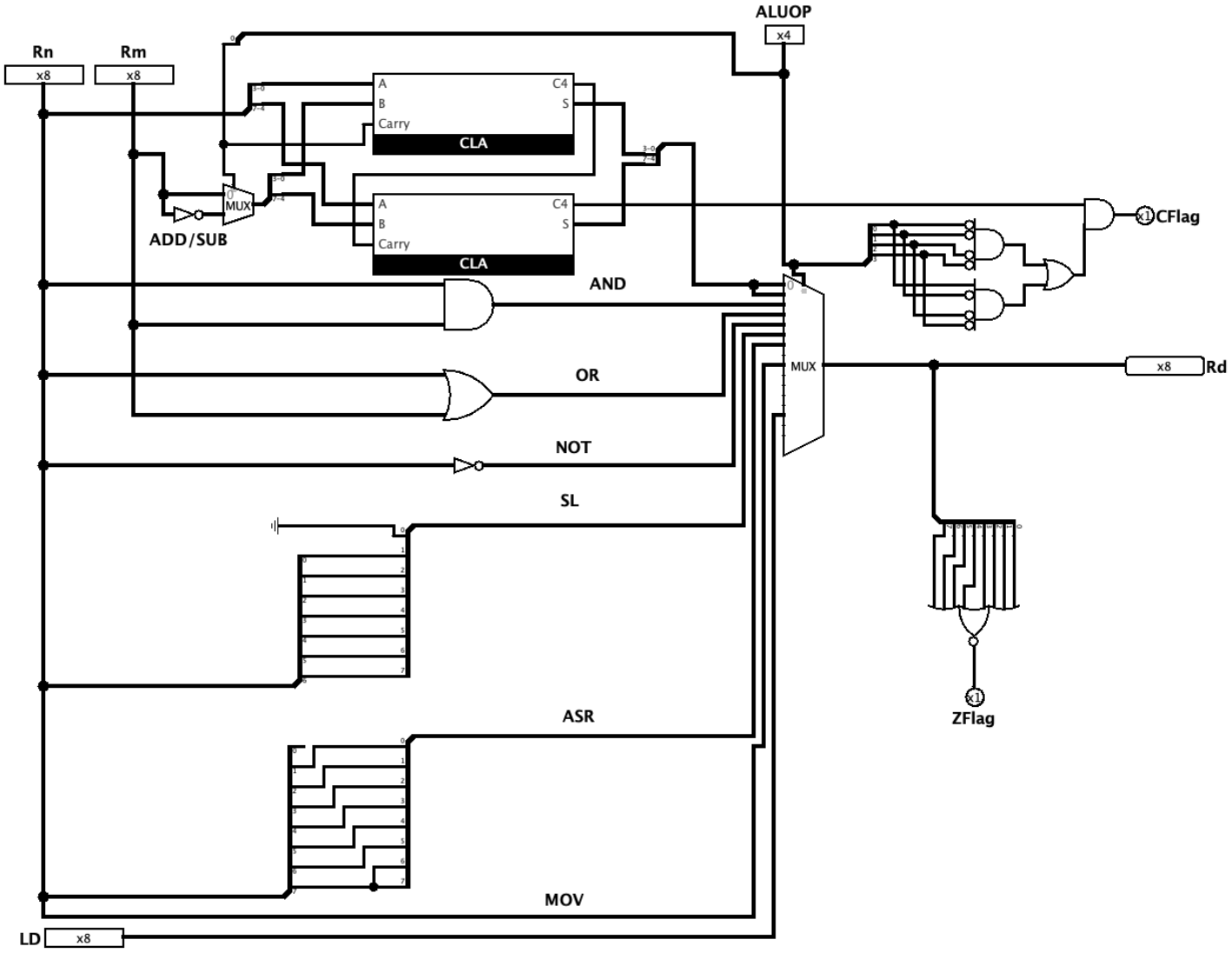


Description:

The controller of MISCSIM is used for the separation of our instruction words and the enabling of certain operations. In the controller, we take the input of the instruction word and move it to the specified format output. We split by the two-bit OP code, the 4 bit ALU OP code, and then the register addresses at the end of the instruction word. There is also a split in order to store the 8 bit immediate value. That can be used by the movement's immediate instruction. The LD and ST outputs act as the write enable and output enable of the RAM, as well as the stores output changing the write, and enable of the register file when needed (see Appendix 6). The controller also takes in the inputs of Zflag and Cflag which are used by the BNZ and BCS instructions. The logic in the controller shown takes these status bits in and changes checks whether not that there can be a loop or not. The two outputs are fed into an OR gate that is outside of this subcircuit.

APPENDIX 4

Arithmetic Logic Unit



Description:

The Arithmetic Logic Unit seen above takes in the input of Rn and RM from the register file and performs all of the arithmetic for the simulator. The input of ALUOP on the top right takes the given operation code and chooses which output is needed to be taken back to the register file. The first two operations are shown at the top as they have codes 0 and 1. There is a mux that is used in order to negate and add a carry for subtraction into two four-bit carry lookahead adders. Under these, there are the rest of the instructions all being fed into the multiplexer for output. At the top right, we have logic that ensures the carry flag is set on the correct ALU operations only. Under that logic, we have a splitter that is used to tell whether or not the result of the last operation is zero or not. We then lead those two flag outputs to a status register that is located in the main circuit (Appendix 6).

APPENDIX 5 (SEE BELOW FOR IMAGE)

DISPLAY DECODER DRIVER

Description:

The display decoder drive that I implemented in my circuit uses the double dabble procedure in order to get the outputs needed for the displaying of the temperatures. On the left-hand side of the circuit, I am checking whether or not the inputted value is a negative value. If it is I am negating it and adding one to it using the Mux in order to choose between the negated or the non-negated format of the number. I am using my CLAs in order to get the twos to complement the version of the number. The double dabble algorithm is an algorithm that uses multiple half-adders in order to shift bits as well as makes these adders conditional on what number they are. From left to right my decoder is adding three and shifting the bits using half adders. There is one input which is the number that is wanted to be displayed and there are 4 outputs such as the sign of the number that is displayed and the respective hundreds, tens, and ones outputs. The sign is grabbed at the beginning in order to ensure that negation does not lose the sign of the number as it is being displayed. The circuit could have been reduced to a smaller size but I followed the constraint which is to not have an additional subcircuit for my half adder. The outputs go to their relative BCD to seven segment displays outside of this subcircuit and in the main circuit (Appendix 6).

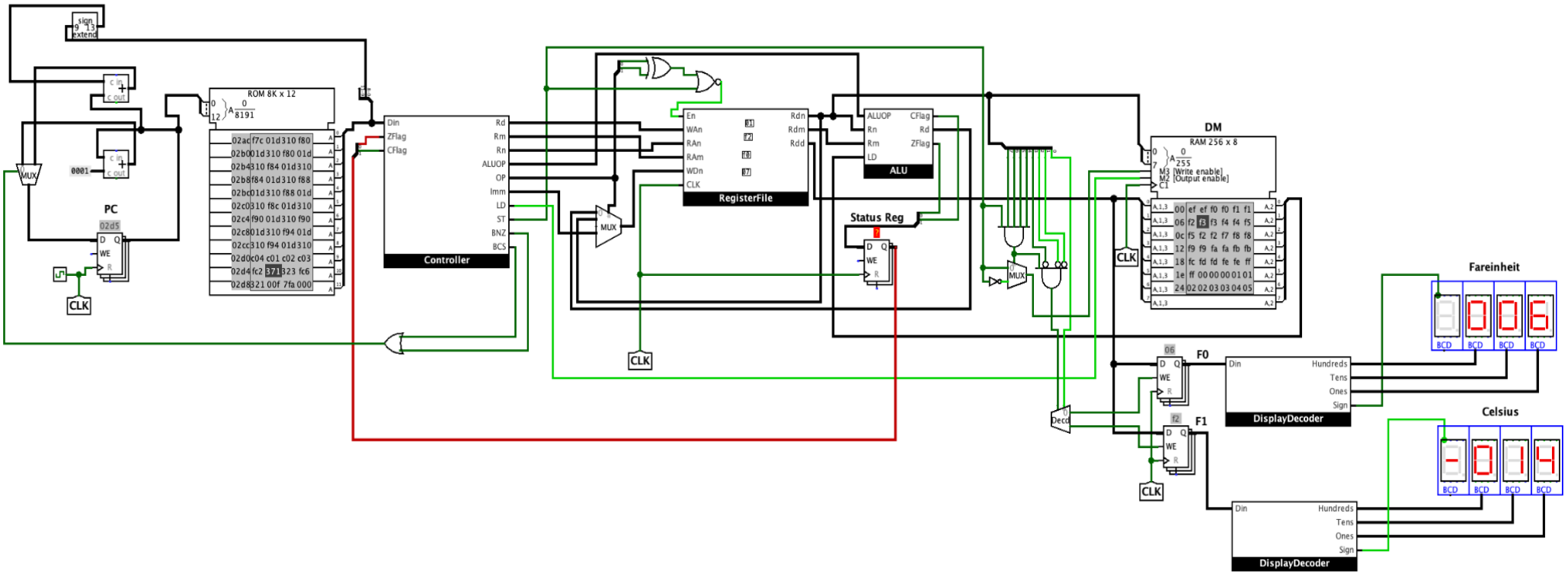

CLA
CLA
Sign
Din
Hundred
Tens
Ones

APPENDIX 6 (SEE BELOW FOR IMAGE)

MAIN

Description:

The image below is of the main circuit and holds all of the logic for the instructions such as BNZ, load, BCS, store, and move immediately. Working from left to right I used two adders in order to choose whether or not to add one or add the target address which is decided by that multiplexer. The line leads to an or gate that is one when a branch instruction is called. The value from ROM is fed into my controller and the bits are split up into the enables that are needed in order to run the correct instruction. Above my register file is the logic that tells whether or not the instruction is a store and it is fed into my enable of my register file which stores the values after every operation. If we follow the ST output label from my controller we can see that it is split up in order to check if we are trying to store into addresses F0 and F1 of our RAM. The mux is used there to negate the input enable if need be. There is a selection like ran to the decoder below to distinguish between the F0 and F1 registers which hold Fahrenheit and Celsius respectively. The LD line is led into the output enable of the ROM and when the output is enabled reads the address given by Rdn and returns the value from ROM at that address. There is a mux just to the right of my controller to distinguish between a regular ALU operation or a move immediate or a simple move. The status registers line is led into my controller to control the branches when the specific parameters are set.


sign extend
9 13
C in
C out
0001
MUX
PC
02d5
D Q
WE
R
CLK
ROM 8K x 12
0
12
A
0
8191
02ac f7c 01d 310 f80
02b0 01d 310 f80 01d
02b4 310 f84 01d 310
02b8 f84 01d 310 f88
02bc 01d 310 f88 01d
02c0 310 f8c 01d 310
02c4 f90 01d 310 f90
02c8 01d 310 f94 01d
02cc 310 f94 01d 310
02d0 c04 c01 c02 c03
02d4 fc2 371 323 fc6
02d8 321 00f 7fa 000
Din
ZFlag
CFlag
Rd
Rm
Rn
ALUOP
OP
Imm
LD
ST
BNZ
BCS
Controller
En
WAn
RAn
RAm
WDn
CLK
01
f2
f0
07
Rdn
Rdm
Rdd
RegisterFile
ALUOP
Rn
Rm
LD
CFlag
Rd
ZFlag
ALU
Status Reg
1
D Q
WE
R
0
MUX
Deco
DM
RAM 256 x 8
0
7
A
0
255
M3 [Write enable]
M2 [Output enable]
C1
A,1,3
A,2
00 ef ef f0 f0 f1 f1
06 f2 f8 f3 f4 f4 f5
0c f5 f2 f2 f7 f8 f8
12 f9 f9 fa fa fb fb
18 fc fd fd fe fe ff
1e ff 00 00 00 01 01
24 02 02 03 03 04 05
06
F0
f2
F1
Din
Hundreds
Tens
Ones
Sign
DisplayDecoder
Fareinheit
0 0 0 6
Celsius
- 0 1 4
BCD

Program 1: Temperature Conversion

Implement a Fahrenheit to Celsius converter program that continuously converts Fahrenheit to Celsius and displays both on the system decimal displays. The display at 0xF0 will display Fahrenheit, and the display at 0xF1 will display Celsius. Your program will start by displaying at zero degrees on the F display and also its converted value on the C display ad will increment the Fahrenheit display by 1 continuously. Note that you only need to allow the program to wrap at 127 to -128. The program will loop through all 255 values of Fahrenheit temperature and then begin again at zero degrees.

Part A: *Your first attempt at this should use the data memory as a translation table. This should simplify the process and allow you to get everything else working. I recommend that you first convert 240 values using the full table. You will have to set up the assembler to fill the table. Most of this can be done with constant loading, however, you may have to use some arithmetic.*

You must include both your assembler source and your Name in your files ***tablef2c.s and tablef2c.hex***

ASSEMBLY CODE DESCRIPTION PART A (SEE FINAL APPENDIX BELOW):

While writing the assembly for the conversion in the data table r1 is used to always hold 1 in order to increment the address of the RAM. R0 is used to hold the immediate value which is the Celsius conversion and then is loaded into the RAM. When decoding the instructions there began to be a pattern that could be followed in order to have each Celsius temperature value loaded at the correct address. Due to the fact that this specific simulator does not read binary decimal values like that of the result of 5/9. Each value is rounded making each Celsius value be stored twice in RAM. However, after 5 cycles, there is an instance of only one number that is displayed at its own unique address in the ram. Therefore the procedure for moving an immediate value into a register and then storing can be looped until we run out of space in our ran. We ran out of space at address 239 because we were using addresses F0 and F1 to store values that needed to be displayed in the main circuit. In the particular code shown in the appendix is used to first store all of the values in their relative addresses and then at the end there is a loop to grab the values in their addresses and load them into the display registers.

Part B: *After you get part A working, you will want to implement the conversion formula in the assembler. One way to get a good handle on this is to do it in a high-level language first but using only 8-bit integer values. You can do this in Java or C fairly easily.*

You must include both your assembler source and your Name in your files ***formulaf2c.s and formulaf2c.hex***

The following requirements apply to both the simulator and the assembler.

ASSEMBLY CODE DESCRIPTION PART B:

I was not able to successfully finish this part of the lab however I feel I have a grasp of how it works. In this assembly code, we are meant to use the constant value and formula in order to display the correct value of Celsius for every Fahrenheit number that is given. In order to do this the Fahrenheit to Celsius conversion formula of °C = (°F − 32) x 5/9 in order to get the correct value stored in RAM. To accomplish this we can load a constant of 32 into a register and leave it there to perform subtractions from the Fahrenheit value. Another register can be loaded with the constant value 5 and then there can be 4 shift rights in order to divide by 8. This value can then be multiplied by the result of Fahrenheit minus thirty-two to get the correct answer. For celsius.

ASSEMBLY CODE APPENDIX

PART A ASSEMBLY CODE

```
                ; Daquan Smith
start:  MVI R3 1        ;move 1 into r3 (incrementer) 110000000111
        MVI R0 -17      ;Loading constants          111110111100
        ST R0 R1        ;Storing constants          001100010000
        MVI R0 -17      ;Repeated Process           111110111100
        ADD R1 R1 R3    ;                           000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -16      ;           111111000000
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -16      ;           111111000000
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -15      ;           111111000100
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -15      ;           111111000100
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -14      ;           111111001000
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -13      ;           111111001100
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -13      ;           111111001100
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -12      ;           111111010000
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -12      ;           111111010000
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -11      ;           111111010100
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -11      ;           111111010100
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -10      ;           111111011000
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -10      ;           111111011000
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -9       ;           111111011100
        ADD R1 R1 R3    ;                   000000011101
        ST R0 R1        ;           001100010000
        MVI R0 -8       ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;16
MVI R0 -8       ;
ADD R1 R1 R3    ;
ST R0 R1        ;17
MVI R0 -7       ;
ADD R1 R1 R3    ;
ST R0 R1        ;18
MVI R0 -7       ;
ADD R1 R1 R3    ;
ST R0 R1        ;19
MVI R0 -6       ;
ADD R1 R1 R3    ;
ST R0 R1        ;20
MVI R0 -6       ;
ADD R1 R1 R3    ;
ST R0 R1        ;21
MVI R0 -5       ;
ADD R1 R1 R3    ;
ST R0 R1        ;22
MVI R0 -5       ;
ADD R1 R1 R3    ;
ST R0 R1        ;23
MVI R0 -4       ;
ADD R1 R1 R3    ;
ST R0 R1        ;24
MVI R0 -3       ;
ADD R1 R1 R3    ;
ST R0 R1        ;25
MVI R0 -3       ;
ADD R1 R1 R3    ;
ST R0 R1        ;26
MVI R0 -2       ;
ADD R1 R1 R3    ;
ST R0 R1        ;27
MVI R0 -2       ;
ADD R1 R1 R3    ;
ST R0 R1        ;28
MVI R0 -1       ;
ADD R1 R1 R3    ;
ST R0 R1        ;29
MVI R0 -1       ;
ADD R1 R1 R3    ;
ST R0 R1        ;30
MVI R0 0        ;
ADD R1 R1 R3    ;
ST R0 R1        ;31
MVI R0 0        ;
ADD R1 R1 R3    ;
ST R0 R1        ;32
MVI R0 0        ;
ADD R1 R1 R3    ;
ST R0 R1        ;33
MVI R0 1        ;
ADD R1 R1 R3    ;
ST R0 R1        ;34
MVI R0 1        ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;35
MVI R0 2        ;
ADD R1 R1 R3    ;
ST R0 R1        ;36
MVI R0 2        ;
ADD R1 R1 R3    ;
ST R0 R1        ;37
MVI R0 3        ;
ADD R1 R1 R3    ;
ST R0 R1        ;38
MVI R0 3        ;
ADD R1 R1 R3    ;
ST R0 R1        ;39
MVI R0 4        ;
ADD R1 R1 R3    ;
ST R0 R1        ;40
MVI R0 5        ;
ADD R1 R1 R3    ;
ST R0 R1        ;41
MVI R0 5        ;
ADD R1 R1 R3    ;
ST R0 R1        ;42
MVI R0 6        ;
ADD R1 R1 R3    ;
ST R0 R1        ;43
MVI R0 6        ;
ADD R1 R1 R3    ;
ST R0 R1        ;44
MVI R0 7        ;
ADD R1 R1 R3    ;
ST R0 R1        ;45
MVI R0 7        ;
ADD R1 R1 R3    ;
ST R0 R1        ;46
MVI R0 8        ;
ADD R1 R1 R3    ;
ST R0 R1        ;47
MVI R0 8        ;
ADD R1 R1 R3    ;
ST R0 R1        ;48
MVI R0 9        ;
ADD R1 R1 R3    ;
ST R0 R1        ;49
MVI R0 10       ;
ADD R1 R1 R3    ;
ST R0 R1        ;50
MVI R0 10       ;
ADD R1 R1 R3    ;
ST R0 R1        ;51
MVI R0 11       ;
ADD R1 R1 R3    ;
ST R0 R1        ;52
MVI R0 11       ;
ADD R1 R1 R3    ;
ST R0 R1        ;53
MVI R0 12       ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;54
MVI R0 12       ;
ADD R1 R1 R3    ;
ST R0 R1        ;55
MVI R0 13       ;
ADD R1 R1 R3    ;
ST R0 R1        ;56
MVI R0 13       ;
ADD R1 R1 R3    ;
ST R0 R1        ;57
MVI R0 14       ;
ADD R1 R1 R3    ;
ST R0 R1        ;58
MVI R0 15       ;
ADD R1 R1 R3    ;
ST R0 R1        ;59
MVI R0 15       ;
ADD R1 R1 R3    ;
ST R0 R1        ;60
MVI R0 16       ;
ADD R1 R1 R3    ;
ST R0 R1        ;61
MVI R0 16       ;
ADD R1 R1 R3    ;
ST R0 R1        ;62
MVI R0 17       ;
ADD R1 R1 R3    ;
ST R0 R1        ;63
MVI R0 17       ;
ADD R1 R1 R3    ;
ST R0 R1        ;64
MVI R0 18       ;
ADD R1 R1 R3    ;
ST R0 R1        ;65
MVI R0 18       ;
ADD R1 R1 R3    ;
ST R0 R1        ;66
MVI R0 19       ;
ADD R1 R1 R3    ;
ST R0 R1        ;67
MVI R0 20       ;
ADD R1 R1 R3    ;
ST R0 R1        ;68
MVI R0 20       ;
ADD R1 R1 R3    ;
ST R0 R1        ;69
MVI R0 21       ;
ADD R1 R1 R3    ;
ST R0 R1        ;70
MVI R0 21       ;
ADD R1 R1 R3    ;
ST R0 R1        ;71
MVI R0 22       ;
ADD R1 R1 R3    ;
ST R0 R1        ;72
MVI R0 22       ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;73
MVI R0 23       ;
ADD R1 R1 R3    ;
ST R0 R1        ;74
MVI R0 23       ;
ADD R1 R1 R3    ;
ST R0 R1        ;75
MVI R0 24       ;
ADD R1 R1 R3    ;
ST R0 R1        ;76
MVI R0 25       ;
ADD R1 R1 R3    ;
ST R0 R1        ;77
MVI R0 25       ;
ADD R1 R1 R3    ;
ST R0 R1        ;78
MVI R0 26       ;
ADD R1 R1 R3    ;
ST R0 R1        ;79
MVI R0 26       ;
ADD R1 R1 R3    ;
ST R0 R1        ;80
MVI R0 27       ;
ADD R1 R1 R3    ;
ST R0 R1        ;81
MVI R0 27       ;
ADD R1 R1 R3    ;
ST R0 R1        ;82
MVI R0 28       ;
ADD R1 R1 R3    ;
ST R0 R1        ;83
MVI R0 28       ;
ADD R1 R1 R3    ;
ST R0 R1        ;84
MVI R0 29       ;
ADD R1 R1 R3    ;
ST R0 R1        ;85
MVI R0 30       ;
ADD R1 R1 R3    ;
ST R0 R1        ;86
MVI R0 30       ;
ADD R1 R1 R3    ;
ST R0 R1        ;87
MVI R0 31       ;
ADD R1 R1 R3    ;
ST R0 R1        ;88
MVI R0 31       ;
ADD R1 R1 R3    ;
ST R0 R1        ;89
MVI R0 32       ;
ADD R1 R1 R3    ;
ST R0 R1        ;90
MVI R0 32       ;
ADD R1 R1 R3    ;
ST R0 R1        ;91
MVI R0 33       ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;92
MVI R0 33       ;
ADD R1 R1 R3    ;
ST R0 R1        ;93
MVI R0 34       ;
ADD R1 R1 R3    ;
ST R0 R1        ;94
MVI R0 35       ;
ADD R1 R1 R3    ;
ST R0 R1        ;95
MVI R0 35       ;
ADD R1 R1 R3    ;
ST R0 R1        ;96
MVI R0 36       ;
ADD R1 R1 R3    ;
ST R0 R1        ;97
MVI R0 36       ;
ADD R1 R1 R3    ;
ST R0 R1        ;98
MVI R0 37       ;
ADD R1 R1 R3    ;
ST R0 R1        ;99
MVI R0 37       ;
ADD R1 R1 R3    ;
ST R0 R1        ;100
MVI R0 38       ;
ADD R1 R1 R3    ;
ST R0 R1        ;101
MVI R0 38       ;
ADD R1 R1 R3    ;
ST R0 R1        ;102
MVI R0 39       ;
ADD R1 R1 R3    ;
ST R0 R1        ;103
MVI R0 40       ;
ADD R1 R1 R3    ;
ST R0 R1        ;104
MVI R0 40       ;
ADD R1 R1 R3    ;
ST R0 R1        ;105
MVI R0 41       ;
ADD R1 R1 R3    ;
ST R0 R1        ;106
MVI R0 41       ;
ADD R1 R1 R3    ;
ST R0 R1        ;107
MVI R0 42       ;
ADD R1 R1 R3    ;
ST R0 R1        ;108
MVI R0 42       ;
ADD R1 R1 R3    ;
ST R0 R1        ;109
MVI R0 43       ;
ADD R1 R1 R3    ;
ST R0 R1        ;110
MVI R0 43       ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;111
MVI R0 44       ;
ADD R1 R1 R3    ;
ST R0 R1        ;112
MVI R0 45       ;
ADD R1 R1 R3    ;
ST R0 R1        ;113
MVI R0 45       ;
ADD R1 R1 R3    ;
ST R0 R1        ;114
MVI R0 46       ;
ADD R1 R1 R3    ;
ST R0 R1        ;115
MVI R0 46       ;
ADD R1 R1 R3    ;
ST R0 R1        ;116
MVI R0 47       ;
ADD R1 R1 R3    ;
ST R0 R1        ;117
MVI R0 47       ;
ADD R1 R1 R3    ;
ST R0 R1        ;118
MVI R0 48       ;
ADD R1 R1 R3    ;
ST R0 R1        ;119
MVI R0 48       ;
ADD R1 R1 R3    ;
ST R0 R1        ;120
MVI R0 49       ;
ADD R1 R1 R3    ;
ST R0 R1        ;121
MVI R0 50       ;
ADD R1 R1 R3    ;
ST R0 R1        ;122
MVI R0 50       ;
ADD R1 R1 R3    ;
ST R0 R1        ;123
MVI R0 51       ;
ADD R1 R1 R3    ;
ST R0 R1        ;124
MVI R0 51       ;
ADD R1 R1 R3    ;
ST R0 R1        ;125
MVI R0 52       ;
ADD R1 R1 R3    ;
ST R0 R1        ;126
MVI R0 52       ;
ADD R1 R1 R3    ;
ST R0 R1        ;127
MVI R0 -88      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-128
MVI R0 -88      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-127
MVI R0 -87      ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;-126
MVI R0 -87      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-125
MVI R0 -86      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-124
MVI R0 -86      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-123
MVI R0 -85      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-122
MVI R0 -85      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-121
MVI R0 -84      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-120
MVI R0 -83      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-119
MVI R0 -83      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-118
MVI R0 -82      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-117
MVI R0 -82      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-116
MVI R0 -81      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-115
MVI R0 -81      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-114
MVI R0 -80      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-113
MVI R0 -80      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-112
MVI R0 -79      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-111
MVI R0 -78      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-110
MVI R0 -78      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-109
MVI R0 -77      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-108
MVI R0 -77      ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;-107
MVI R0 -76      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-106
MVI R0 -76      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-105
MVI R0 -75      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-104
MVI R0 -75      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-103
MVI R0 -74      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-102
MVI R0 -73      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-101
MVI R0 -73      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-100
MVI R0 -72      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-99
MVI R0 -72      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-98
MVI R0 -71      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-97
MVI R0 -71      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-96
MVI R0 -70      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-95
MVI R0 -70      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-94
MVI R0 -69      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-93
MVI R0 -68      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-92
MVI R0 -68      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-91
MVI R0 -67      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-90
MVI R0 -67      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-89
MVI R0 -66      ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;-88
MVI R0 -66      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-87
MVI R0 -65      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-86
MVI R0 -65      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-85
MVI R0 -64      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-84
MVI R0 -63      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-83
MVI R0 -63      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-82
MVI R0 -62      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-81
MVI R0 -62      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-80
MVI R0 -61      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-79
MVI R0 -61      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-78
MVI R0 -60      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-77
MVI R0 -60      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-76
MVI R0 -59      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-75
MVI R0 -58      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-74
MVI R0 -58      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-73
MVI R0 -57      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-72
MVI R0 -57      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-71
MVI R0 -56      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-70
MVI R0 -56      ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;-69
MVI R0 -55      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-68
MVI R0 -55      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-67
MVI R0 -54      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-66
MVI R0 -53      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-65
MVI R0 -53      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-64
MVI R0 -52      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-63
MVI R0 -52      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-62
MVI R0 -51      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-61
MVI R0 -51      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-60
MVI R0 -50      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-59
MVI R0 -50      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-58
MVI R0 -49      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-57
MVI R0 -48      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-56
MVI R0 -48      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-55
MVI R0 -47      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-54
MVI R0 -47      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-53
MVI R0 -46      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-52
MVI R0 -46      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-51
MVI R0 -45      ;
```

```
ADD R1 R1 R3    ;
ST R0 R1        ;-50
MVI R0 -45      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-49
MVI R0 -44      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-48
MVI R0 -43      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-47
MVI R0 -43      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-46
MVI R0 -42      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-45
MVI R0 -42      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-44
MVI R0 -41      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-43
MVI R0 -41      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-42
MVI R0 -40      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-41
MVI R0 -40      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-40
MVI R0 -39      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-39
MVI R0 -38      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-38
MVI R0 -38      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-37
MVI R0 -37      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-36
MVI R0 -37      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-35
MVI R0 -36      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-34
MVI R0 -36      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-33
MVI R0 -35      ;
ADD R1 R1 R3    ;
ST R0 R1        ;-32
MVI R0 -35      ;
```

```
        ADD R1 R1 R3    ;
        ST R0 R1        ;-31
        MVI R0 -34      ;
        ADD R1 R1 R3    ;
        ST R0 R1        ;-30
        MVI R0 -33      ;
        ADD R1 R1 R3    ;
        ST R0 R1        ;-29
        MVI R0 -33      ;
        ADD R1 R1 R3    ;
        ST R0 R1        ;-28
        MVI R0 -32      ;
        ADD R1 R1 R3    ;
        ST R0 R1        ;-27

        MVI R0 1        ;          110000000100 = C04  holds 1 for increment
        MVI R1 240      ;           111111000001 = FC1  holds address
        MVI R2 0        ;          110000000010 = c02  holds the display addresses
        MVI R3 0        ;          110000000011 = c03  holds hex value from address

tempdisplay:
        MVI R2 240      ;           1111110000
        LD R1 r3
        ST R3 R2
        MVI R2 241
        ST R1 R2
        ADD R3 R0 R3
        BNZ tempdisplay ;
end:    BNZ end         ;
```