Digital Electronics & Computer Engineering (E85)

Harris Spring 2019

Final Exam

This is a closed-book take-home exam. Electronic devices including calculators are not allowed, except on the computer question on the last page. You are permitted two 8.5x11" sheets of paper with notes.

You are bound by the HMC Honor Code while taking this exam.

The first part of the exam is written, while the final page is done on the computer based on your E85 Lab 11. The entire Lab 11 that you use (including datapath and controller) must be your own work; it cannot, for example, include somebody else's controller. The exam is intended to be doable in 3 hours if you have prepared adequately. However, there will be no limit on the time you are allowed except that the written portion must be completed in one contiguous block of time and the computer part must be completed in another contiguous block of time. A contiguous block of time is a period of time working at a desk without breaking for meals, naps, socializing, etc. You cannot study for E85 or consult E85 resources between the two blocks of time. Please manage your time wisely and do not let the exam expand to take more time than is justified.

Return the exam to the E85 box in the Engineering Department Office no later than Wednesday 5/15 at noon (5/10 at 5 pm for seniors).

Alongside each question, the number of points is written in brackets. All work and answers should be written directly on this examination booklet, except for printouts. Use the backs of pages if necessary. Write neatly; illegible answers will be marked wrong. Show your work for partial credit.

Name: _____

Do Not Write Below This Point

Page 2:	_____	/ 5
Page 3:	_____	/ 6
Page 4:	_____	/ 6
Page 5:	_____	/ 7
Page 6:	_____	/ 7
Page 7:	_____	/ 8
Page 12:	_____	/ 6
Total:	_____	/ 45

[2] Consider a 512-word x 64-bit RAM. The RAM requires k bits of address and 2^s bit cells total. What are k and s?

k: _____

s: _____

[3] An IEEE half-precision floating point number is similar to an ordinary floating-point number but has 10 bits of significand and 5 bits of exponent with a bias of 15.

Write -6.625 as a half-precision floating point number, and express your answer in hexadecimal.

Number: _____

How many Logic Elements does each of the following Verilog modules require to fit onto your Cyclone V FPGA? Explain your reasoning.

```
module xor7(input  logic [6:0] a,
            output logic       y);
  assign y = a[0] ^ a[1] ^ a[2] ^ a[3] ^ a[4] ^ a[5] ^ a[6];
endmodule
```

[2] Logic Elements: _____

```
module lfsr(input  logic clk,
            input  logic reset,
            output logic q);
  logic state[9:0];
  always_ff @(posedge clk, posedge reset)
    if (reset) state <= 9'b0;
    else state <= {state[8:0], state[8]^state[4]};
endmodule
```

[2] Logic Elements: _____

```
module cmddec(input  logic [24:21] instr,
              output logic isAND, isXOR, isSUB, isADD, isCMP);
  always_comb
    begin
      isAND = 0; isXOR = 0; isSUB = 0; isADD = 0; isCMP = 0;
      case (instr)
        4'b0000: isAND = 1;
        4'b0001: isXOR = 1;
        4'b0010: isSUB = 1;
        4'b0100: isADD = 1;
        4'b1010: isCMP = 1;
      endcase
    end
endmodule
```

[2] Logic Elements: _____

[6] Write an assembly language program to find the position of the most significant '1' bit in a 32-bit word in R1. Your answer should be in the range of 0 (a '1' only in the least significant bit) to 31 (a 1 in the most significant bit). For example, if the word is

0000 0000 0000 0000 0000 0000 0001 0100

the answer should be 4, because bits 4 and 2 are both '1' and bit 4 is the most significant. If the word is all 0's, return 32. Return your result in R0.

The following program is supposed to sum an array of 10 ints. The base address of the array is in R4, the sum is in R6, and the loop counter is in R5. The result is in R6.

```
          MOV R5, #0
          MOV R6, #0
LOOP
          LDR R0, [R4, R5]
          ADD R6, R6, R0
          ADD R5, R5, #1
          CMP R5, #10
          BNE LOOP
DONE
```

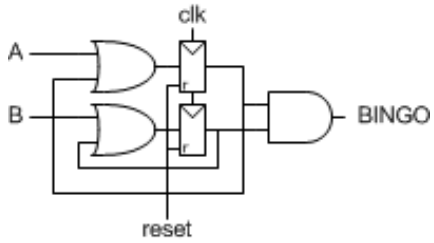
[2] The program has one bug. Explain what the bug is.

[2] Mark up the code to fix the bug with as little change to the program as possible.

[3] How many cycles will the program take to run on a pipelined processor? Assume the processor has the same hazards as the pipelined processor in class, but that it has been enhance all of the instructions/modes needed for this program? Define the number of cycles required to include fetching all of the instructions, but not waiting for the last instructions in the pipeline to complete after the program reaches DONE. The number of cycles should be the same in your fixed code as in the original; if not, count cycles based on the original code. Explain your reasoning.

Cycles _____

Consider the following circuit.



[2] Draw a state transition diagram corresponding to the circuit.

[2] Give a simple explanation of when the circuit asserts BINGO.

[3] Write behavioral (not structural) Verilog code gracefully describing the circuit.

```
module final(input  logic clk, reset,
             input  logic A, B,
             output logic BINGO);









endmodule
```

The ARM `LDRB Rd, [Rn, imm]` is like LDR but loads a single byte into the bottom 8 bits of a register and fills the upper 24 bits with zeros. Recall that ARM is a little-endian architecture in which the least significant byte of a word is stored at the lowest address in the word. Modify the ARM multicycle processor to support the LDRB instruction, using as little additional hardware as feasible.

[3] Mark up the attached multicycle processor diagram and ALU to handle the new instructions.

[2] Mark up the attached multicycle controller (including state transition diagram and truth tables) to handle the new instructions.

[2] The attached multicycle memfile.s test code has highlighted modifications to test the new instruction. As compared to the memfile.s from Lab 11, it changes the constant in the ADDLT command at 0x40 from #1 to #255 and replaces the LDR instruction at 0x4C with an LDRB.

Translate the LDRB command to machine language. Express your code in hexadecimal. The format for a memory instruction is given below. The cond field for ALWAYS is 1110. The six control bits are described in the table below.

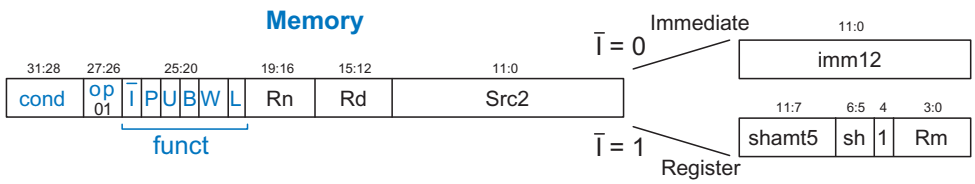


Table 6.9 Offset type control bits for memory instructions

Bit	Meaning Ī	Meaning U
0	Immediate offset in Src2	Subtract offset from base
1	Register offset in Src2	Add offset to base

Table 6.10 Index mode control bits for memory instructions

P	W	Index Mode
0	0	Post-index
0	1	Not supported
1	0	Offset
1	1	Pre-index

Table 6.11 Memory operation type control bits for memory instructions

L	B	Instruction
0	0	STR
0	1	STRB
1	0	LDR
1	1	LDRB

LDRB R2, [R0, #237] _____

[1] Predict what value should be written to mem[248] at the last line of the program.

Predicted Value: _____

Multicycle Processor

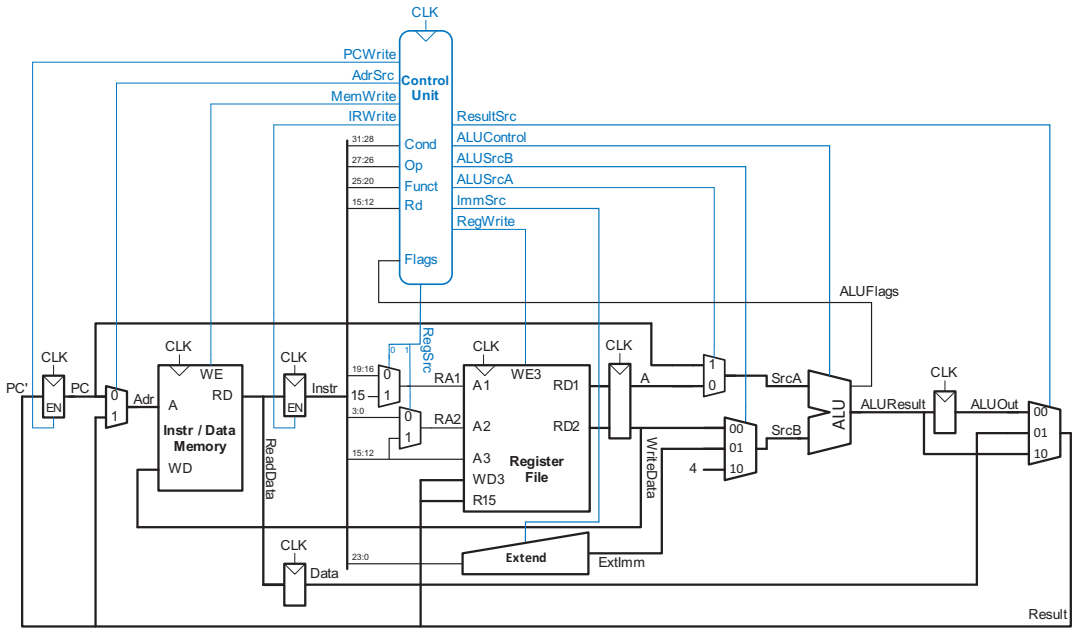
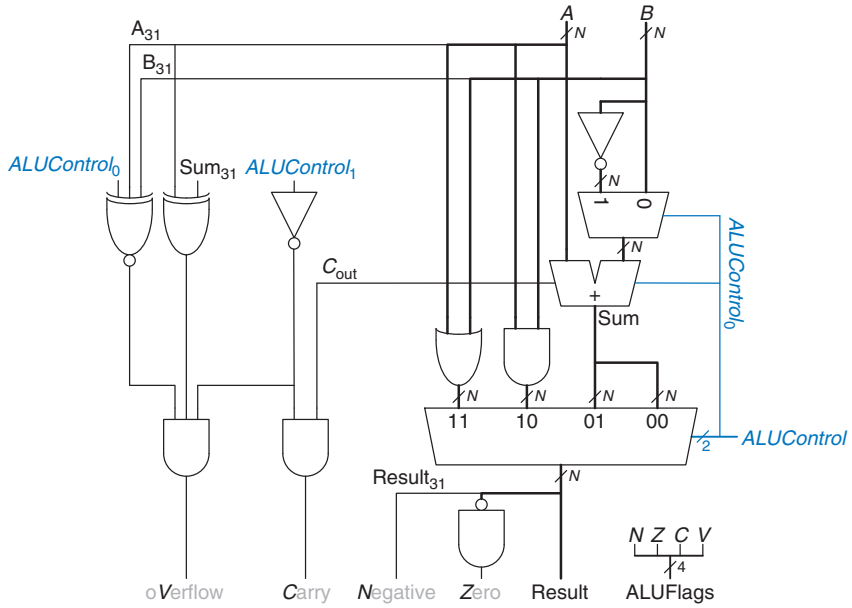


Figure 7.30 Complete multicycle processor

ALU



Multicycle Controller

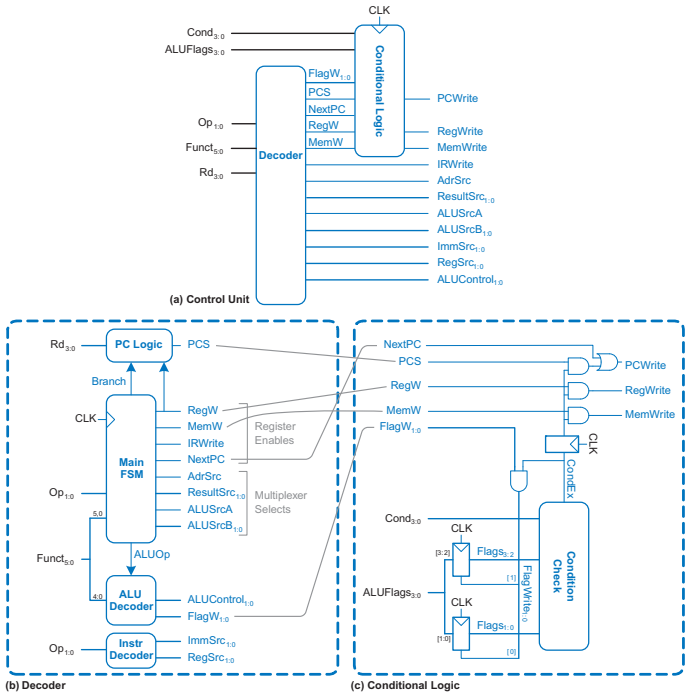


Figure 7.31 Multicycle control unit

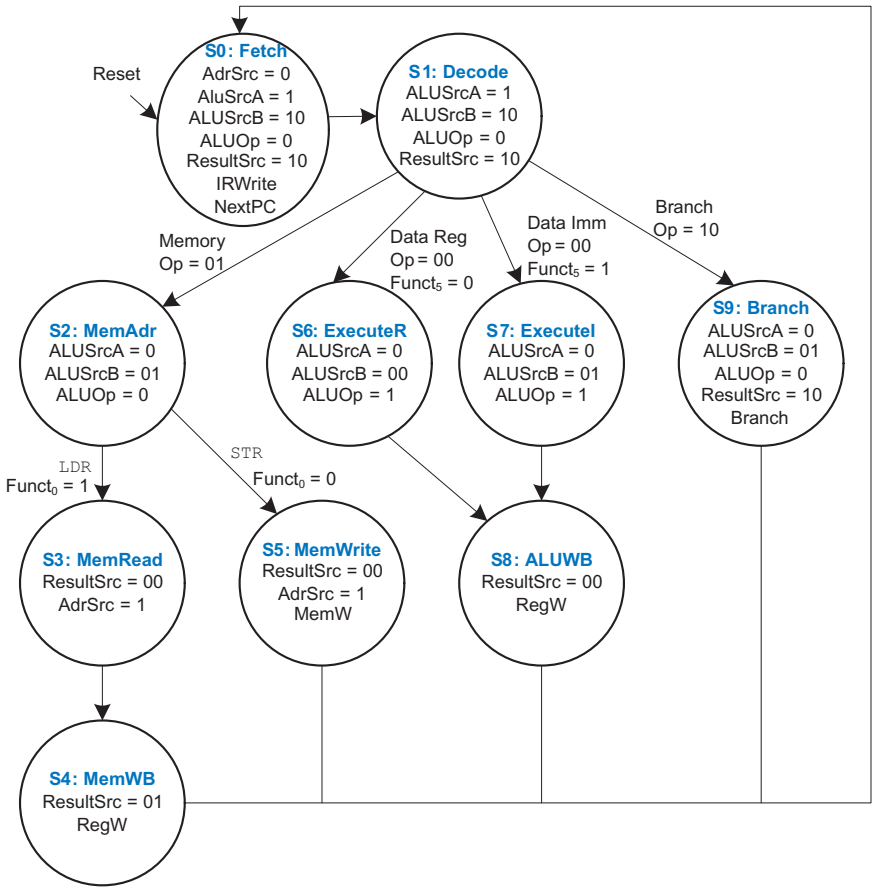


Table 7.6 Instr Decoder logic for *RegSrc* and *ImmSrc*

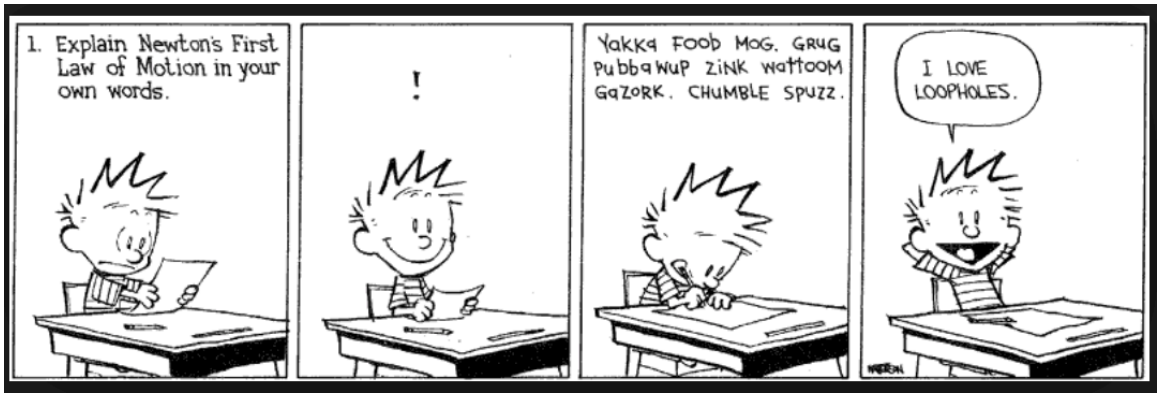
Instruction	Op	$Funct_5$	$Funct_0$	$RegSrc_1$	$RegSrc_0$	$ImmSrc_{1:0}$
LDR	01	X	1	X	0	01
STR	01	X	0	1	0	01
DP immediate	00	1	X	X	0	00
DP register	00	0	X	0	0	00
B	10	X	X	X	1	10

$ALUOp$	$Funct_{4:1}$ (cmd)	$Funct_0$ (S)	Type	$ALUControl_{1:0}$	$FlagW_{1:0}$
0	X	X	Not DP	00 (Add)	00
1	0100	0	ADD	00 (Add)	00
		1			11
	0010	0	SUB	01 (Sub)	00
		1			11
	0000	0	AND	10 (And)	00
		1			10
	1100	0	ORR	11 (Or)	00
		1			10

```
        ; memfile.dat
MAIN
SUB R0, R15, R15       ; R0 = 0             1110 000 0010 0 1111 0000 0000 0000 1111 E04F000F 0x00
ADD R2, R0, #5          ; R2 = 5            1110 001 0100 0 0000 0010 0000 0000 0101 E2802005 0x04
ADD R3, R0, #12        ; R3 = 12            1110 001 0100 0 0000 0011 0000 0000 1100 E280300C 0x08
SUB R7, R3, #9         ; R7 = 3             1110 001 0010 0 0011 0111 0000 0000 1001 E2437009 0x0c
ORR R4, R7, R2         ; R4 = 3 OR 5 = 7    1110 000 1100 0 0111 0100 0000 0000 0010 E1874002 0x10
AND R5, R3, R4         ; R5 = 12 AND 7 = 4  1110 000 0000 0 0011 0101 0000 0000 0100 E0035004 0x14
ADD R5, R5, R4         ; R5 = 4 + 7 = 11    1110 000 0100 0 0101 0101 0000 0000 0100 E0855004 0x18
SUBS R5, R5, #10       ; R5 = 11 - 10 = 1   1110 001 0010 1 0101 0101 0000 0000 1010 E255500A 0x1c
SUBSGT R5, R5, #2      ; R5 = 1 - 2 = -1    1100 001 0010 1 0101 0101 0000 0000 0010 C2555002 0x20
ADD R5, R5, #12        ; R5 = -1 + 12 = 11  1110 001 0100 0 0101 0101 0000 0000 1100 E285500C 0x24
SUBS R8, R5, R7        ; R8 = 11 - 3 = 8    1110 000 0010 1 0101 1000 0000 0000 0111 E0558007 0x28
BEQ END                ; not taken          0000 1010 0000  0000 0000 0000 0000 1100 0A00000F 0x2c
SUBS R8, R3, R4        ; R8 = 12 - 7  = 5   1110 000 0010 1 0011 1000 0000 0000 0100 E0538004 0x30
BGE AROUND             ; should be taken    1010 1010 0000  0000 0000 0000 0000 0000 AA000000 0x34
ADD R5, R0, #0         ; should be skipped  1110 001 0100 0 0000 0101 0000 0000 0000 E2805000 0x38
AROUND
SUBS R8, R7, R2        ; R8 = 3 - 5 = -2    1110 000 0010 1 0111 1000 0000 0000 0010 E0578002 0x3c
ADDLT R7, R5, #255     ; R7 = 11+255=266    1011 001 0100 0 0101 0111 0000 1111 1111 B28570FF 0x40
SUB R7, R7, R2         ; R7 = 266-5= 261    1110 000 0010 0 0111 0111 0000 0000 0010 E0477002 0x44
STR R7, [R3, #224]     ; mem[12+224] = 261    1110 010 1100 0 0011 0111 0000 0101 0100 E58370E0 0x48
LDRB R2, [R0, #237]     ;
ADD R15, R15, R0       ; PC <- PC + 8       1110 000 0100 0 1111 1111 0000 0000 0000 E08FF000 0x50
ADD R2, R0, #14        ; shouldn't happen   1110 001 0100 0 0000 0010 0000 0000 0001 E280200E 0x54
B END                  ; always taken       1110 1010 0000  0000 0000 0000 0000 0001 EA000001 0x58
ADD R2, R0, #13        ; shouldn't happen   1110 001 0100 0 0000 0010 0000 0000 0001 E280200D 0x5C
ADD R2, R0, #10         ; shouldn't happen  1110 001 0100 0 0000 0010 0000 0000 0001 E280200A 0x60
END
STR R2, [R0, #248]     ; mem[248] = ?       1110 010 1100 0 0000 0010 0000 1111 1000 E58020F8 0x64
```

END OF WRITTEN PORTION OF EXAM

DO NOT PROCEED PAST THIS POINT UNTIL YOU ARE PREPARED TO CEASE ALL WORK ON THE WRITTEN PORTION AND MOVE ON TO THE COMPUTER PORTION.



COMPUTER PORTION OF EXAM

Once you start this question, you may refer to the written portion of the exam, but may not spend any more time on the written portion or change any of your answers on that portion.

Modify your ARM multicycle processor from Lab 11 to support the LDRB instruction. Modify your memfile.dat to replace the existing LDR instruction with the LDRB and to change the ADDLT instruction's highlighted in the earlier code.

[2] Print out your Verilog code and circle or highlight the lines you modified.

[4] Print out a simulation waveform showing at least the value being written to memory location 248 on the last cycle. Circle this value in the waveform.