TIME: 11:09
11:23
14 min

14*4= 56 min
⇒ ok length

University of Victoria
Faculty of Engineering
Department of Computer Science
CSc 230 Computer Architecture and Assembly Language

MIDTERM EXAM October 19, 2015

NAME: LillAnne. (Print)

ID NUMBER: ____________________

SIGNATURE: ____________________

TIME: 75 minutes
INSTRUCTOR: LillAnne Jackson

Question	Value	Mark
1	1	
2	~~6~~ 4	
3	4	
4	4	
5	2	
6	1	
7	2	
8	6	
9	6	
10	8	
TOTAL	~~40~~ 38	

109 in class
1 excused late
6 RCSD
116.

121 = Registered.

STUDENTS MUST CHECK THE NUMBER OF PAGES IN THIS EXAMINATION PAPER BEFORE BEGINNING TO WRITE, AND REPORT ANY DISCREPANCY IMMEDIATELY TO THE INVIGILATOR

- The questions are to be answered on the examination paper.
- The exam is **NOT** open book. Calculators are **NOT** permitted.
- A formula for a calculation is acceptable anywhere a calculation might be required.
- A copy of the *AVR Instruction Set Summary* will be provided with this exam.
- The marks assigned to each question are printed within square brackets.
- There are 8 pages in this document, including this cover page.
- It is strongly recommended that you read the entire exam through from beginning to end before beginning to answer the questions.

1. **[1 mark]** What is the major distinction between the von Neumann and Harvard architectures?

von Neuman - One bank of memory addresses.
- shared by code + data.

Harvard - Separate addresses for code memory + data memory.

2. **[4 marks]** Given two hexadecimal numbers, state their corresponding values :
   a. in binary
   b. as a decimal, if the number is an unsigned integer

Hex	Binary (8 bits):	Decimal Integer: unsigned
$3E_{16}$	$0011\ 1110_2$	$3*16 + 14 = 62_{10}$
$B8_{16}$	$1011\ 1000_2$	$11*16 + 8 = 184_{10}$.

3. **[4 marks]** Given two positive decimal numbers which are in the range of 8-bit binary numbers:
   a. provide the binary equivalent of each number
   b. provide the 2's Complement

Decimal	Binary (8 bits):	2's Complement	
41_{10}	0010 1001 (32 16 8)		positive, so same as binary
103_{10}	0110 0111 (64 32)		positive, so same as binary

4. **[4 marks]** Given the two signed 8-bit binary numbers and their sum below, what would be the value that would be in the C (carry),V (overflow), Z (zero), and N (negative) bits of a typical (i.e., AVR's) Status Register immediately after the sum is created?

	Values in Status Register (SR)
0110 1110 0101 0111 1100 0101	C 0 V 1 Z 0 N 1

5. **[2 marks]** Given the two 8-bit binary numbers below, perform bitwise AND and OR operations (on each bit in the numbers).

0110 1110 AND 0101 0111 01000110	0110 1110 OR 0101 0111 0111 1111

6. **[1 mark]** In a system with 24 address lines, what is the address space? (Ie, What is the maximum number of locations that can be addressed in this system?)

2^{24} ; from 0 to $2^{24}-1$

(An expression or formula, rather than the calculated result is acceptable.)

7. **[2 marks]**

a. What is the output of an assembler program?

binary numbers that represent maching language

b. What does a two-pass assembler produce on each of its two passes?

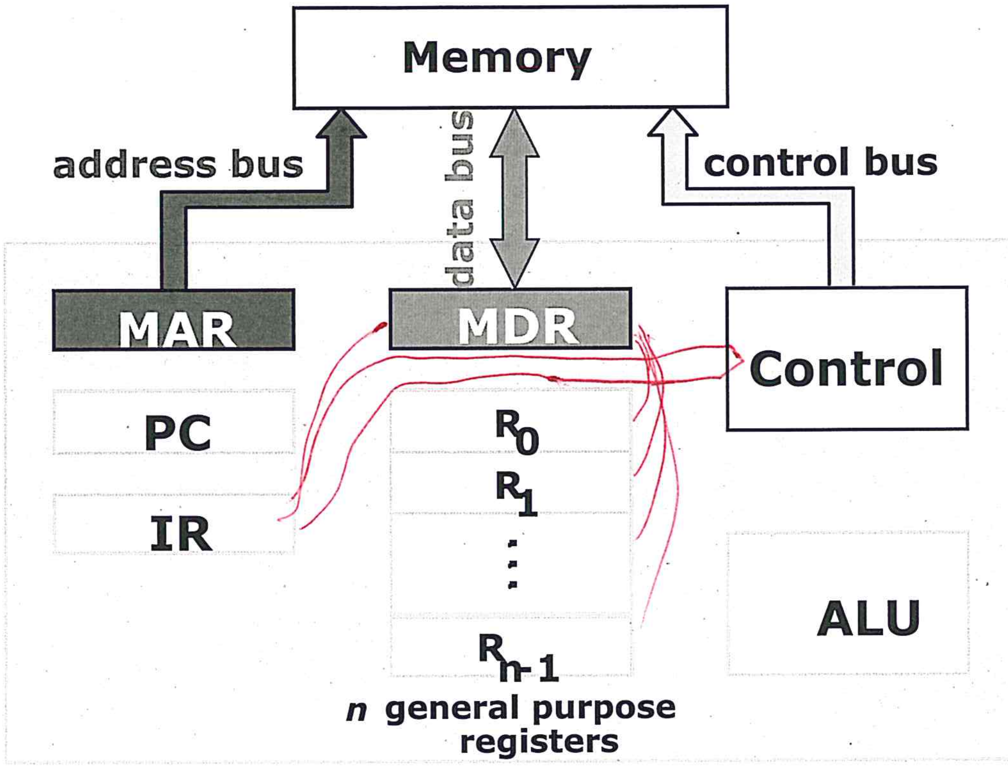
pass 1: a table of the labels + their corresponding addresse

pass 2: The machine language; including the resolution of labels to numbers

8. **[6 marks]** Consider the diagram below of a (generic) CPU with its internal datapath connections. Use the diagram to give the *detailed steps of the Fetch and Decode/Execute cycles* of the instruction: **`LD R0,0x021F`**

Note: The instruction uses *direct* addressing.

↳ The address of the data is in the instructio



Fetch:

MAR ← PC

address bus ← MAR

Control bus ← read signal

<<wait for Data>>.

data bus ← memory (instruction)

MDR ← data bus

IR ← MDR

PC update:

ALU ← PC

ALU (add).

PC ← ALU

<The next page is provided to give additional space for you to write this answer>

Decode

control ← IR

<<Control unit determines opcode + operands>>.

Execute

2nd fetch (Required!)

MAR ← 0x021F (from operands of instruction)

address bus ← MAR

control bus ← read signal

<< wait for Data>>.

data bus ← memory (Data from address 012F)

MDR ← data bus

Execute

R0 ← MDR.

9. **[6 marks]** Given the picture of the system's registers, including some initial values in those registers, and some of its memory (below) determine the values in the registers after the execution of the following assembly language code.

```
        LDI    R26, 00   ; X = R27:R26  (High):(Low)
        LDI    R27, 02
        LD     R0, X
        INC    R26
        CP     R0, R1
        BREQ   skip
        ADD    R2, R1
skip:   LSL    R3
```

R0	0x~~00~~ 10	10
R1	0x2f	2F
R2	0x~~10~~ 3F	3F
R3	0x1f 0001 1111 0011 1110 = 3E	3E
R26	0x~~10~~ ~~00~~ 01	~~00~~ 01
R27	0x~~10~~ 02	02

Memory Addresses	Memory Content
0x0200	10
0x0201	2f
0x0202	10
0x0203	fd
0x0204	06
0x0205	00
0x0206	0f
0x0207	de

0001 1111
0011 1110
3 E

Final Values in Registers:

R0	0x10	R1	0x2F	R26	0x01
R2	0x3F	R3	0x3E	R27	0x02

10. **[8 marks]** Complete the AVR assembly language program below. It must calculate the sum of the finite geometric series:

$$\sum_{k=0}^{7} 2^k = 2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7$$

And store the result in the memory (RAM) location called `sum`. Pseudo-code is provided as an algorithm that describes how to complete the code.

```
.include "m2560def.inc"
;This program calculates the sum of a finite geometric series.
;Pseudo- code:
;       sum = 0
;       term = 1
;       for( index = 0; index < size; index++) {
;               sum = sum + term
;               term = term * 2
;       }
; Registers use:
;       r16 <-> sum
;       r17 <-> term
;       r18 <-> loop index
;       r19 <-> size, i.e., number of terms
;       r__<-> ________________(fill in if other register used)
;       r__<-> ________________(fill in if other register used)
;       r__<-> ________________(fill in if other register used)


.cseg

.def    sum=r0 16
.def    term=r16 7
.def    index=r2 18
.def    terms=r3 19
        size
```

;Continued on next page

```
;initializing the registers
clr sum            ; sum = 0
ldi term, 0x01     ; term = 1

;for( index = 0; index < size; index++) {
;       sum = sum + term
;       term = term * 2
;}
;** Insert your code here: perform the for loop above **
```

```
        ldi   size,0x08        -1
        clr   index            -1

for:    cp    index, size      }-1
        breq  ~~endfor~~ ~~done~~ doTheStore            0,1,2,3,4,5,6,7
                                                          8
        add   sum, term        -1
        lsl   term             -2         add term,term
        inc   index            -1
        rjmp  for              -1
```

```
doTheStore  ;storing the result in dseg location called sum
        ldi r27,HIGH(sum)
        ldi r26,Low(sum)
        st  X,sum


done: jmp done
;End of code segment


; Data Storage area
.dseg
.org  0x200
sum   .byte  2
```

The End