Midterm Exam 13 December 2018, questions and answers

Microprocessing System (Đại học Quốc gia Thành phố Hồ Chí Minh)

HCMIU
Instructor: Hồ Trung Mỹ

Subject: Micro-processing systems
Solution to Sample Midterm Exam AY1819-S1 – Class: EE – Duration: 90 minutes

INSTRUCTIONS:

This is an (open book, closed laptop and closed cellphone) exam and you have 90 minutes to complete it. The exam is worth 30% of your final mark. Please do your answers on the question papers.

Note: We use AVR ATmega32 microprocessor in this exam.

Q1. (60 pts) Given an assembly progran as follows:

```
.include "m32def.inc"
.def temp = R16
.def  a = R17
.def  b = R18
.def Sum_L = R19
.def Sum_H = R20
.def Counter = R21
.equ K1 = $A4
.equ K2 = $76
.equ data_addr = $110
.cseg
.org 0
main:  LDI    a,K1 | K2          ; #1
       LDI    b,low(K1 * K2)     ; #2
       ADD    a,b                ; #3
       CPI    a,K2               ; #4
       BREQ   Here               ; #5
       LDI    XL,low(data_addr)
       LDI    XH,high(data_addr)
       CLR    Sum_L
       CLR    Sum_H
       LD     Counter,X+         ; #6
Loop:  LD     temp,X+            ; #7
       ADD    Sum_L,temp
       BRCC   Next
       INC    Sum_H
Next:  DEC    Counter
       BRNE   Loop               ; #8
       LDI    YL,low(result)
       LDI    YH,high(result)
       STD    Y+1,Sum_H
       ST     Y,Sum_L
Here:  RJMP   Here               ; #9
.dseg
.org $130
result: .byte 2
```

a) (15 pts) Fill in the blanks: (number values in hexadecimal and flags in binary)

- After the execution of the instruction with comment "#1", R17 = **$F6**
- After the execution of the instruction with comment "#2", R18 = **$98**
- After the execution of the instruction with comment "#3": (Assume the initial value of SREG is 0)
  - R17 = **$8E**
  - Flags: H = **0**, S = **1**, Z = **0**, N = **1**, V = **0**, C = **1**,

b) (10 pts) Specify the addreesing mode for each following instruction:

- The instruction with comment "#3": **Two Register direct**
- The instruction with comment "#4": **Single Register with Immediate**
- The instruction with comment "#9": **Relative Program Addressing**

c) (10 pts) For the following instruction sequence (with comments "#4' and "#5"), mark with an √ the conditional branch instructions which would transfer control to Here if used in place of BREQ

Instruction sequence	BRCC	BRCS	BRLO	BRSH	BRLT	BRGE
`CPI a,K2` `BREQ Here`	√			√	√	

d) (20 pts) Assume that before the execution of the instruction with comment "#6":

D($0110) = $03; D($0111) = $E8; D($0112) = $DC; D($0113) = $C9;

We consider iterations of the loop (from the instruction with comment "#7" to the instruction with comment "#8"). If any data changes in general registers, indicate that:

	R16	R19	R20	R21	R26	R27
Before the loop	$00	$00	$00	$03	$11	$01
After the iteration #1	$E8	$E8		$02	$12	
After the iteration #2	$DC	$C4	$01	$01	$13	
After the iteration #3	$C9	$8D	$02	$00	$14	

Before the execution of the instruction with comment "#9":

D($0130) = $8D and D($0131) = $02

e) (5 pts) Rewrite the following instruction sequence without using "STD"

Using "STD"	Without using "STD"
STD Y+1,Sum_H ST Y,Sum_L	ST Y+,Sum_L ST Y,Sum_H

Q2. (10 pts) Write an assembly program that checks continuously the status of PA7 (MSB of Port A) and does the following operation:

If PA7 is one then Port B = min(Port C, Port D), otherwise Port B = (Port C – Port D) x 2

Ans.

```
.include "m32def.inc"
.def   temp = R16
.def   temp_C = R17
.def   temp_D = R18
.cseg
.org 0
; I/O configuration
        CBI    DDRA,7
        CLR    temp
        LDI    DDRC,temp
        LDI    DDRD,temp
        SER    temp
        LDI    DDRB,temp
Loop:
        IN     temp_C,PINC
        IN     temp_D,PIND
        SBIS   PINA,7
        RJMP   Otherwise
        CP     temp_C,temp_D
        BRLO   Next
        MOV    temp_C,temp_D
Next:
        OUT    PORTB,Temp_C
        RJMP   Loop
Otherwise:
        SUB    temp_C,temp_D
        LSL    temp_C
        RJMP   Next
```

Q3. (20 pts)

a) (15 pts) Write a subroutine **COMP_U8** (comparison of two **unsigned 8 bit numbers**) with the following operation table:

R21	Operation	Comment
< R20	R22 = 4xR20 + R21/4	Use shift instructions for division and multiplication
= R20	R22 = abs(R20 – 17)	Absolute value of difference of R20 and 17
> R20	R22 = max(R20, R21)	

Ans.

```
; comparison of unsigned 8-bit numbers
COMP_S8:    MOV   R22, R20
            CP    R21, R20
            BRSH  Case_GE

Case_LT:    ; Case 1:     R21 < R20
            LSL   R22     ; 2R20
            LSL   R22     ; 4R20
            LSR   R21     ; R21/2
```

```
            LSR    R21     ; R21/4
            ADD    R22, R21
            RET
Case_GE:    ;  R21 >= R20
            BRNE    Case_GT
Case_EQ:    ;Case 2:        R21 = R20
            SUBI   R22, 17
            BRPL  Continue
            NEG    R22     ; R22 = –R22
Continue:   RET
Case_GT:    ;Case 3:        R21 > R20      => max is R21
            MOV    R22,R21
            RET
```

b) (5 pts) Write a program segment that reads data from from Port B and places it in R20, reads data from Port C and places it in R21, then calls subroutine **COMP_U8**, and sends data from R22 to SRAM location \$70.

Ans.

```
.include "M32DEF.INC"
.org 0
            LDI     R16, HIGH(RAMEND)
            OUT     SPH, R16
            LDI     R16, LOW(RAMEND)
            OUT     SPL, R16; SP = RAMEND
            CLR     R16
            OUT     DDRB, R16               ; Port B: Input port
            OUT     DDRC, R16               ; Port C: Input port
            IN      R20, PINB
            IN      R21, PINC
            CALL    COMP_U8
            STS     $70, R22
 HERE:      RJMP    HERE
```

Q4. (10 pts) Implement the following algorithm of the subroutine **Div_AB** that divides the unsigned value of the A by the unsigned value of the B. The resulting quotient is placed in the A and the remainder is placed in the B. We use AVR register R21 for A, AVR register R22 for B, and AVR register R23 for C (temporary variable in the algorithm).

<table>
<tr><th>Pseudo code</th><th>Your AVR assembly code</th></tr>
<tr><td><pre>
begin
C ← 0
While (A >= B)
{
   A ← A – B
   C ← C + 1
}
B ← A
A ← C
return
</pre></td><td><pre>
.def   A = R21
.def   B = R22
.def   C = R23
Div_AB:
            CLR   C   ; or LDI   C,0
Loop:
            CP   A, B
            BRLO   Finish ; if A < B goto Finish
            SUB   A, B
            INC   C
            RJMP   Loop
Finish:
            MOV   B, A
            MOV   A, C
            RET
</pre></td></tr>
</table>

End of question paper

Explanations for some answers

Q1.

a)

```
main:  LDI    a,K1 |  K2          ; #1  R17 = K1 OR K2
       LDI    b,low(K1 * K2)  ; #2  R18 = low byte of (K1 x K2)
       ADD    a,b             ; #3  R17 = R17 + R18
```

➢ AVR assembler will evaluate the expression K1 & K2 and low(K1 * K2) before compiling code:

$A4	|	1	0	1	0	0	1	0	0
$76		0	1	1	1	0	1	1	0
$F6		1	1	1	1	0	1	1	0

K1 * K2 = $A4 x $76 = $4B98 ⇒ low(**K1 * K2**) = low byte of ($4B98) = $98

⇒ **R17 = $F6 and R18 = $98**

➢ ADD a.b ⇒ R17 = R17 + R18 = **$F6** + $98 = $8E with C = 1

	1	1	1	1	0	0	0	0	0
$F6	+	1	1	1	1	0	1	1	0
$98		1	0	0	1	1	0	0	0
$F6		1	0	0	1	0	1	1	0

H = 0, N = 1, C = 1 ⇒ V = C ⊕ C_{MSB} = 1 ⊕ 1 = 0 ⇒ S = N ⊕ V = 1 ⊕ 0 = 1

Therefore, we have

- After the execution of the instruction with comment "#1", R17 = **$F6**
- After the execution of the instruction with comment "#2", R18 = **$98**
- After the execution of the instruction with comment "#3": (Assume the initial value of SREG is 0)
  - R17 = **$8E**
  - Flags: H = _0_, S = _1_, Z = _0_, N = _1_, V = _0_, C = _1_,

c) We have: a =R17 = $8E , and K2 = $76

➢ Comparison instruction does the operation ***operand1 – operand2*** for udating status flags of SREG. If we focus on only C and Z flags, we have the function table as follows:

operand1 – operand 2	**C**	**Z**
< 0 (or operand1 < operand2)	**1**	**0**
= 0 (or operand1 = operand2)	**0**	**1**
> 0 (or operand1 >operand2)	**0**	**0**

In our case:

operand1 = $8E > operand2 = $76 ⇒ C = 0 ⇒ true condition for BRCC and BRSH

For signed number comparison:

operand1 = $8E = –114 < operand2 = $76 = +118 ⇒ true condition BRLT

(or quick comparsion: $8E is negative and $76 is positive without hex to decimal conversion)

Therefore, we can checks for BRCC, BRSH and BRLT:

Instruction sequence	**BRCC**	**BRCS**	**BRLO**	**BRSH**	**BRLT**	**BRGE**
`CPI a,K2` `BREQ Here`	√			√	√	

d)

Given data memory:

Address	Contents
$0110	$03
$0111	$E8
$0112	$DC
$0113	$C9

```
LDI   XL,low(data_addr)         ;  XL = R26 = low($110) = $10
LDI   XH,high(data_addr)        ;  XH = R27 = high$110) = $01
. . .
LD    Counter,X+                ; #6  R21 = D(X) = D($0110) = $03 and X =X +1 = $0111
```

So, before the loop:

R21 = $03 (number of array elements), R26 = $11, and R27 = $01

➢ Analysis of the loop: (before the loop: X = $0111)

	Iteration #1	Iteration #2	Iteration #3
Loop: LD temp,X+	R16=D($0111)= $E8 X= X + 1 = $0112	R16=D($0112)= $DC X= X + 1 = $0113	R16=D($0113)= $C9 X= X + 1 = $0114
ADD Sum_L,temp	R19= 0 + $E8 = $E8 and C =0	R19= $E8+$DC=$C4 and C =1	R19= $C4+$C9=$8D and C =1
BRCC Next	C=0 ⇒ Branch to Next	C = 1 ⇒ next inst.	C = 1 ⇒ next inst.
INC Sum_H		R20= R20+1=0+1= 1	R20= R20+1=1+1= 2
Next: DEC Counter	R21=$03–1=$02 and Z=0	R21=$02–1=$01 and Z=0	R21=$01–1= 0 and Z=1
BRNE Loop	Z= 0 ⇒ Branch to Loop	Z=0⇒Branch to Loop	Z = 1 ⇒ next inst.

Therefore, we have the following table:

	R16	R19	R20	R21	R26	R27
Before the loop	$00	$00	$00	$03	$11	$01
After the iteration #1	$E8	$E8		$02	$12	
After the iteration #2	$DC	$C4	$01	$01	$13	
After the iteration #3	$C9	$8D	$02	$00	$14	

e) Given Y = $0130

```
STD     Y+1,Sum_H ; D(Y+1) = D($0130+1) = D($0131) = R20 = $02 and Y is unchaged!
ST      Y,Sum_L   ; D(Y) = D($0130) = D($0131) = R19 = $8D
```

We have the same result with the instruction sequence:

```
ST      Y,Sum_L
STD     Y+1,Sum_H
```

Without using "STD":

```
ST      Y+,Sum_L; D(Y) = D($0130) = D($0131) = R19 = $8D and Y = Y+1 = $0131
ST      Y,Sum_H; D(Y) = D($0131) = R20 = $02
```

Q2.

If PA7 is one then Port B = min(Port C, Port D), otherwise Port B = (Port C – Port D) x 2

Pesudo code:

```
If    PA7 = 1 then
      If   Port C < Port D then  PortB = PortC  else  PortB = PortD;
Else
      Port B = (Port C – Port D) x 2;
```

Note: We use temp_C (R17) and temp_D (R18) for holding the current input values of Port C and Port D (i.e. temp_C = PINC and temp_D = PIND).

Pseudo code	***AVR assembly code***	
	Solution 1	***Solution 2***

If PA7 = 1 then
<operations #1>
Else
<operations #2>
	SBIS PINA,7
	RJMP Else_part
	; then part
	<operations #1>
	RJMP Continue
	Else_part:
	<operations #2>
	Continue:
		SBIC PINA,7
		RJMP Then_part
		; Else part
		<operations #2>
		RJMP Continue
		Then_part:
		<operations #2>
		Continue:

// PortB = min(PortC, PortD)
If Port C < Port D then
PortB = PortC
else
PortB = PortD
	CP temp_C,temp_D
	BRLO C_LT_D
	; temp_C >= temp_D
	OUT PORTB, temp_D
	RJMP Continue
	C_LT_D:
	OUT PORTB, temp_C
	Continue:
		CP temp_C,temp_D
		BRSH C_GE_D
		; temp_C < temp_D
		OUT PORTB, temp_C
		RJMP Continue
		C_GE_D:
		OUT PORTB, temp_D
		Continue:

temp_C = min(temp_C,temp_D)	CP temp_C,temp_D BRLO Continue ; temp_C >= temp_D MOV temp_C, temp_D Continue:	CP temp_D,temp_C BRSH Continue ; temp_C > temp_D MOV temp_C, temp_D Continue:

Port B = (Port C – Port D) x 2;	SUB temp_C,temp_D
	LSR temp_C
	OUT PORTB, temp_C
		SUB temp_C,temp_D
		ADD temp_C,temp_C
		OUT PORTB, temp_C

For Port B = min(Port C, Port D), we can rewrite with the way as follows:

```
            CP      temp_C,temp_D
            BRLO    Next
            ;  temp_C >= temp_D
            MOV       temp_C, temp_D
Next:       ; temp_C = min (temp_C, temp_D)
            OUT     PORTB, temp_C
```

Q4.

Pseudo code	***AVR assembly code***

While (A >= B)
{
<operations>
}
	While_Loop:
	CP A,B
	BRLO Exit_While ; exit when A < B is true
	; or BRCS Exit_While
	<operations>
	RJMP While_Loop
	Exit_While:

Note:
- Exit condition = NOT(while condition) = NOT(A >= B) = A < B
- Read “Tutorial #2” for more implementations of control structures!