ชื่อ-สกุล.....เลขประจำตัว.....section.....

มหาวิทยาลัยสงขลานครินทร์
คณะวิศวกรรมศาสตร์

สอบกลางภาค: ภาคการศึกษาที่ 2 **ปีการศึกษา:** 2553

วันที่สอบ: 26 ธันวาคม 2553 **เวลาสอบ:** 13.30-16.30 น.

รหัสวิชา: 240-305 **ห้องสอบ:** 5817

ชื่อวิชา: Microprocessor Architectures and the Assembly Language

อ่านรายละเอียดของข้อสอบ และคำแนะนำให้เข้าใจก่อนเริ่มทำข้อสอบ

เวลา: 3 ชั่วโมง (180 นาที)

รายละเอียดของข้อสอบ: ข้อสอบมีทั้งหมด 10 หน้า

- เป็นกระดาษคำถามจำนวน 7 หน้า(รวมปก)
- เป็น Datasheet จำนวน 3 หน้า

อนุญาต: เครื่องเขียนต่างๆ เช่น ปากกา หรือดินสอ

ไม่อนุญาต: เครื่องคิดเลข หนังสือและสมุดจด และกระดาษโน้ตใดๆ เข้าห้องสอบ

คำสั่ง:

- ให้ทำทุกข้อ
- คำตอบทั้งหมดจะต้องเขียนลงในข้อสอบ
- เขียนชื่อและรหัสให้ชัดเจนในข้อสอบ <u>**ทุกแผ่น**</u> แผ่นใดไม่เขียนหรือเขียนไม่ครบจะถูกตัดคะแนนแผ่นละ 1 คะแนน
- คำตอบส่วนใดอ่านไม่ออก จะถือว่าคำตอบนั้นผิด
- อ่านคำสั่งเพิ่มเติมในแต่ละข้อให้ชัดเจน

คำแนะนำ:

- โจทย์แต่ละข้อ หากต้องการดูข้อมูลรายละเอียดของคำสั่งและข้อมูลรีจิสเตอร์ของ AVR ให้ดูข้อมูลใน Datasheet ท้ายข้อสอบประกอบซึ่งอาจารย์ได้ใส่ข้อมูลเพิ่มเติมที่จำเป็นในการทำข้อสอบไว้ครบถ้วนแล้ว

ชื่อ-สกุล.....เลขประจำตัว.....section.....

1. อธิบายหน้าที่และการประยุกต์ใช้งานของ Status Register แต่ละบิตของ AVR อย่างละเอียด (5 คะแนน)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2. ตอบคำถามต่อไปนี้ (5 คะแนน)

- ยกตัวอย่างสถาปัตยกรรมไมโครคอนโทรลเลอร์มาอย่างน้อย 3 สถาปัตยกรรม.....

.....

- Watchdog timer มีประโยชน์อย่างไร.....

.....

.....

- วงจร Power-on Reset ทำหน้าที่อะไร.....

.....

.....

- DMA คืออะไร มีประโยชน์อย่างไรในระบบคอมพิวเตอร์..... .....

.....

.....

.....

-Flash memory, SRAM และ EEPROM ต่างกันอย่างไร..... .....

.....

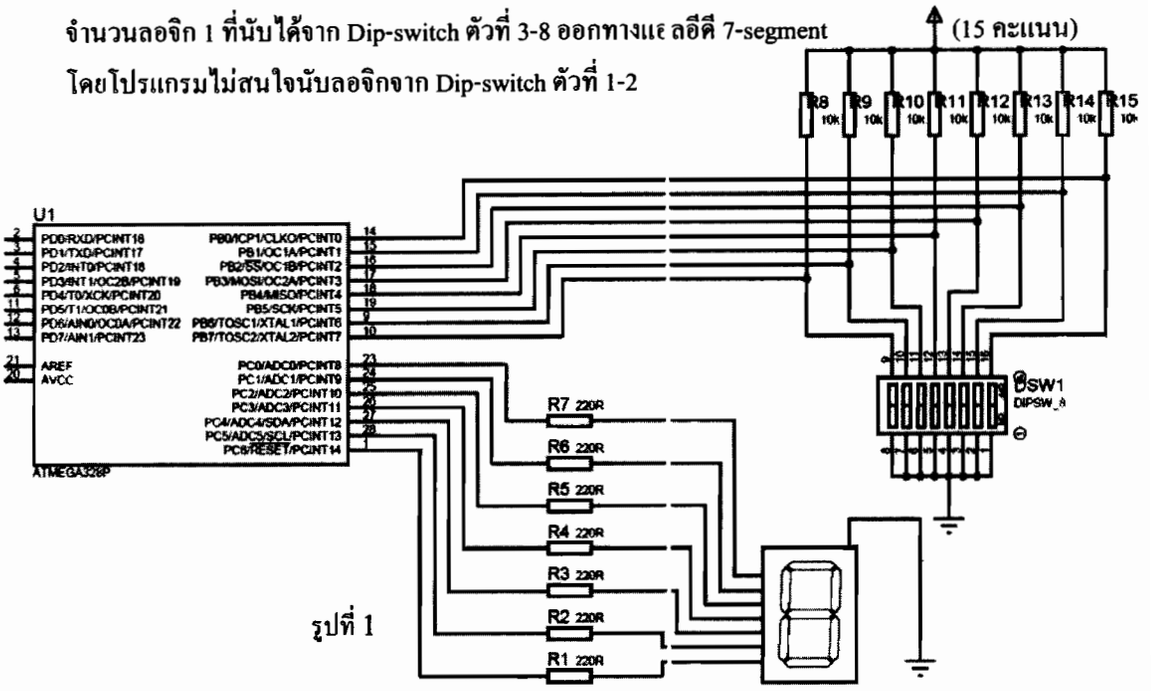
.....

.....

.....

ชื่อ-สกุล.....เลขประจำตัว.....section.....

3. จงเขียนโปรแกรมภาษาแอสเซมบลี้ของ AVR เพื่อทำการรับค่าจาก Dip-switch ดังรูปที่ 1 และแสดงผลจำนวนลอจิก 1 ที่นับได้จาก Dip-switch ตัวที่ 3-8 ออกทางแอลอีดี 7-segment โดยโปรแกรมไม่สนใจนับลอจิกจาก Dip-switch ตัวที่ 1-2 (15 คะแนน)



รูปที่ 1

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ชื่อ-สกุล.....เลขประจำตัว.....section.....

ชื่อ-สกุล.....เลขประจำตัว.....section.....

4. หากจะนำ AVR ซึ่งเป็นซีพียูขนาด 8 bit มาทำการคูณเลขจำนวนเต็มขนาด 32 บิตซึ่งได้ผลลัพธ์ขนาด 64 บิต จะสามารถทำได้อย่างไร จงอธิบายแนวคิดในการเขียนโปรแกรมโดยไม่ต้องเขียนโปรแกรมภาษาแอสเซมบลี้จริงๆ (10 คะแนน)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ชื่อ-สกุล.....เลขประจำตัว.....section.....

5. จงเขียนโปรแกรมภาษาแอสเซมบลี้ของ AVR ซึ่งมีการทำงานเทียบเท่ากับภาษาซีต่อไปนี้ (15 คะแนน)

`void main(void)`
`{`
`  unsigned char a, b,c;`
`  c = find_max(a,b);`
`  c = c-100;`
`}`
	`unsigned char`
	`find_max(unsigned char x,`
	`  unsigned char y)`
	`{`
	`  x = x+5;`
	`  y = y+5;`
	`  if (x>y)`
	`  return x;`
	`  else`
	`  return y;`
	`}`

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ชื่อ-สกุล.....เลขประจำตัว.....section.....

6. จงเขียนโปรแกรมภาษาแอสเซมบลี้ของ AVR ซึ่งมีการทำงานเทียบเท่าภาษาซีต่อไปนี้ (10 คะแนน)

```
unsigned char i, b, c;
unsigned char  a[100];
for (i=0;i<100;i++)
{
   if (a[i] >b)
         a[i] = b;
   else
         a[i] = c;
}
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....ขอให้ทุกคนโชคดีในการทำข้อสอบครับ.....

ชื่อ-สกุล.....เลขประจำตัว.....section.....

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF − Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 − Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd − 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP[1]	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL[1]	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	(Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd − Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd − Rr − C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd − K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	(Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	(Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	(P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	(P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	(SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	(SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	(Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	(Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	(C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	(C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	(C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	(C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	(N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	(N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	(N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	(N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	(H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	(H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	(T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	(T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	(V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	(V = 0) then PC ← PC + k + 1	None	1/2

ชื่อ-สกุล.....เลขประจำตัว.....section.....

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0...6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3...0)←Rd(7...4),Rd(7...4)←Rd(3...0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2

ชื่อ-สกุล.....เลขประจำตัว.....section.....

7	6	5	4	3	2	1	0	
I	T	H	S	V	N	Z	C	SREG
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Syntax:

```
LABEL:  .BYTE expression
```

Example:

```
.DSEG
        var1:   .BYTE 1          ; reserve 1 byte to var1
        table:  .BYTE tab_size   ; reserve tab_size bytes
.CSEG
        ldi     r30,low(var1)    ; Load Z register low
        ldi     r31,high(var1)   ; Load Z register high
        ld      r1,Z             ; Load VAR1 into register 1
```

Syntax:

```
.DEF Symbol=Register
```

Example:

```
.DEF temp=R16
.DEF ior=R0
.CSEG
.ORG 0x67
      ldi     temp,0xf0     ; Load 0xf0 into temp register
      in      ior,0x3f      ; Read SREG into ior register
      eor     temp,ior      ; Exclusive or temp and ior
```

7 0	Addr.	
R0	0x00	
R1	0x01	
R2	0x02	
...		
R13	0x0D	
R14	0x0E	
R15	0x0F	
R16	0x10	
R17	0x11	
...		
R26	0x1A	X-register Low Byte
R27	0x1B	X-register High Byte
R28	0x1C	Y-register Low Byte
R29	0x1D	Y-register High Byte
R30	0x1E	Z-register Low Byte
R31	0x1F	Z-register High Byte