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

มหาวิทยาลัยสงขลานครินทร์

คณะวิศวกรรมศาสตร์

สอบกลางภาค: ภาคการศึกษาที่ 2 **ปีการศึกษา:** 2554

วันที่สอบ: 20 ธันวาคม 2554 **เวลาสอบ:** 13.30-16.30 น.

รหัสวิชา: 241-210 **ห้องสอบ:** R200, S-201, S-101 หัว 4 ม.

ชื่อวิชา: Microprocessor Architectures and the Assembly Language

อ่านรายละเอียดของข้อสอบ และคำแนะนำให้เข้าใจก่อนเริ่มทำข้อสอบ

เวลา: 3 ชั่วโมง (180 นาที)

รายละเอียดของข้อสอบ: ข้อสอบมีทั้งหมด 9 หน้า

ไม่อนุญาต: กระดาษโน้ต สมุดจด หนังสือ และเครื่องคิดเลข

คำสั่ง:

- **<u>ให้ทำทุกข้อ</u>** คำตอบทั้งหมดจะต้องเขียนลงในข้อสอบ
- ห้ามหยิบยืมสิ่งใดๆ ทั้งสิ้นจากผู้อื่น เว้นแต่ผู้คุมสอบจะหยิบยืมให้
- ห้ามนำส่วนหนึ่งส่วนใดของข้อสอบออกจากห้องสอบ
- ผู้ประสงค์จะออกจากห้องสอบก่อนหมดเวลาสอบ **<u>แต่ต้องไม่น้อยกว่า 30 นาที</u>** ให้ยกมือขออนุญาตจากผู้คุมสอบก่อนจะลุกจากที่นั่ง
- เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
- เขียนชื่อ, รหัสและหมายเลข Section ให้ชัดเจนในข้อสอบ **<u>ทุกแผ่น</u>** แผ่นใดไม่เขียนหรือเขียนไม่ครบจะถูกตัดคะแนนแผ่นละ 1 คะแนน
- คำตอบส่วนใดอ่านไม่ออก จะถือว่าคำตอบนั้นผิด
- อ่านคำสั่งเพิ่มเติมในแต่ละข้อให้ชัดเจน

ทุจริตในการสอบ โทษขั้นต่ำคือ ปรับตกในรายวิชาที่ทุจริต และพักการเรียน 1 ภาคการศึกษา

ผู้ออกข้อสอบ

.....

ดร. ปัญญยศ ไชยกาฬ

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

1. จงอธิบายความหมายของนิยามต่อไปนี้ (5 คะแนน)

Mnemonic.....

.....

Bus.....

.....

Instruction Fetching.....

.....

Instruction Register.....

.....

Assembler.....

.....

Subroutine.....

.....

Opcode.....

.....

Operand.....

.....

2. กำหนดให้ก่อนรันโปรแกรมหน่วยความจำ SRAM ของ AVR ตำแหน่ง 0x100-0x103 มีค่า 0xF0, 0xF2, 0x37, 0x51 ตามลำดับ จงแสดงค่าในหน่วยความจำและรีจิสเตอร์หลังรันโปรแกรมต่อไปนี้ (5 คะแนน)

```
        ldi   ZH, 0x01
        ldi   ZL, 0x00
        ld    r1, Z+
        ld    r2, Z+
        ld    r3, -Z
        add   r3, r1
        brcs  AAA
        sbc   r1, r2
AAA:    st    -Z, r1
        st    Z+,r2
```

r1=.....mem[0x100].....

r2=.....mem[0x101].....

r3=.....mem[0x102].....

ZH=.....mem[0x103].....

ZL=.....

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

3. จงกาเครื่องหมาย ✖ หน้าคำสั่งที่<u>ไม่ใช่</u>คำสั่งที่ถูกต้องของซีพียู AVR (5 คะแนน)

.....ADD	R21, R21
.....INC	PORTC
.....PUSH	DDRC
.....POP	DDRD
.....TST	R16, R24
.....PUSH	R16, R20
.....ADC	R26, 30
.....MUL	R30, R31
.....ADD	ZL, ZH
.....COM	PORTB
.....LPM	R30, Z

4. จงตอบคำถามต่อไปนี้ (7 คะแนน)

- Flash memory ต่างจาก EEPROM อย่างไร.....
.....
- กำหนดให้ ซีพียู ATmega88A มี SRAM ภายในเท่ากับ 1 กิโลไบต์ ค่าของรีจิสเตอร์ SP ของซีพียู ดังกล่าวหลังการรีเซ็ตมีค่าเป็นเท่าใด.....
- ค่าระดับแรงดันเลี้ยงซีพียู มีผลต่อความถี่ในการทำงานของ AVR อย่างไร.....
.....
- จงบอกหน้าที่การทำงานของวงจร Watchdog timer.....
.....
- จงบอกข้อดีข้อเสียของสร้างความถี่สัญญาณนาฬิกาของ AVR ด้วยวิธีการต่อ Crystal Oscillator....
.....
- การใช้วงจร Power-on reset ภายในซีพียู AVR มีข้อดีข้อเสียอย่างไร.....
.....
- ไฟล์ที่ได้จากการแอสเซมเบลอร์ภาษาแอสเซมบลี้ของ AVR มีนามสกุลอะไร.....
- DMA คืออะไร มีประโยชน์อย่างไรในระบบคอมพิวเตอร์.....
.....
- ยกตัวอย่างสถาปัตยกรรมไมโครคอนโทรลเลอร์มาอย่างน้อย 4 สถาปัตยกรรม.....
.....
- เพราะเหตุใดจึงกล่าวว่า AVR เป็นซีพียูขนาด 8 บิต.....
.....
- รีจิสเตอร์ DDRC ทำหน้าที่อะไรในซีพียู AVR.....
.....

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

5. จงเขียนโปรแกรมย่อย (Subroutine) ของ AVR ด้วยภาษาแอสเซมบลี้ซึ่งเทียบเท่าการทำงานของภาษาซีต่อไปนี้ (6 คะแนน)

```
signed char find_max(signed char a, signed char b)
{
        if (a>b)
                return a;
        else
                if (a==b)
                        return 0;
                else
                        return b;
}
```

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

6. จงหาค่าในรีจิสเตอร์และหน่วยความจำของซีพียู AVR หลังจากเสร็จสิ้นการรันโปรแกรมภาษาแอสเซมบลี้ต่อไปนี้ (10 คะแนน)

```
.cseg
.org 0x00
                rjmp    RESET
RESET:          ldi     ZH, high(TABLE1)
                ldi     ZL, low(TABLE1)
                ldi     R21, 0xFF
                ldi     R22, 0x0F
                ldi     R23, 0x55
                ldi     R24, 0x7E
                ldi     R25, 0xF0
                ldi     R26, 0x3D
                mov     R27, R25

                st      Z+, R21
                st      Z+, R22
                st      -Z, R23
                st      -Z, R25
                push    R25
                push    R26
                eor     R22, R21
                or      R23, R21
                and     R24, R21
                neg     R25
                com     R26
                pop     R28
                pop     R29

.dseg
.org 0x504
TABLE1:         .byte 20
```

R30=.....

R31=.....

R21=.....

R22=.....

R23=.....

R24=.....

R25=.....

R26=.....

R27=.....

R28=.....

R29=.....

SRAM[0x504].....

SRAM[0x505].....

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

7. จงเขียนโปรแกรมภาษาแอสเซมบลี้เพื่ออ่านค่าจากสวิตช์จำนวน 4 บิตซึ่งต่อกับพอร์ต C บิตที่ 3-6 (PC3-PC6) และนำค่าที่ได้แปลงเป็นเลขฐานสองขนาด 8 บิต แสดงผลออกทางแอลอีดี 7 เซกเมนต์ชนิด Common Anode ที่ต่ออยู่กับพอร์ต B (10 คะแนน)

-------End of Examination------

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	None	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	if (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	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (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.....

Mnemonics	Operands	Description	Operation	Flags	#Clocks
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

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

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
```

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
```