✅ 1) ถ้า “ไม่มีเน็ต” เปิดโปรแกรม รัน ได้ไหม?

ได้ปกติ 100%

สิ่งที่ใช้ในการรันคือ

- compiler / assembler
- device pack (ATmega328P etc.)
- simulator

พวกนี้ติดตั้งไว้ในเครื่องตั้งแต่ตอนลงโปรแกรมแล้ว
👉 ไม่ต้องต่อเน็ต

✅ 2) ถ้า copy project ใส่ Flash Drive ไปเปิดอีกเครื่อง

ทำได้ ถ้าเครื่องนั้นติดตั้ง Microchip Studio แล้ว

เปิดไฟล์พวกนี้ได้เลย:

```
project_name.atsln
project_name.cproj / .asmproj
main.asm
```

แล้วกด Build / Debug ได้ทันที

✅ 3) ค่าที่ตั้งไว้จะตามไปไหม?

ส่วนใหญ่ตามไป ✅

เช่น

- เลือก MCU = ATmega328P
- Optimization
- include file
- memory layout
- debug setting

เพราะมันเก็บในไฟล์ project

⚠️ แต่มีบางอย่างที่ “อาจ” ไม่ตาม

ถ้าอีกเครื่อง

❌ ไม่มี device pack
❌ version เก่ามาก
❌ ไม่มี simulator

อาจ build ไม่ได้

👉 แต่อาจารย์ในห้องสอบปกติเตรียมไว้แล้ว

✅ 4) สิ่งที่ต้องระวังที่สุดเวลาไปเปิดเครื่องใหม่

เปิด solution ให้ถูกไฟล์

เปิดตัวนี้ดีที่สุด (ไฟล์เต่าทอง):

`xxx.atsln`

ไม่ใช่เปิดเฉพาะ main.asm

✅ 5) PICSimLab ต้องมีเน็ตไหม?

ไม่ต้อง
มันเป็น simulator local

ขอแค่

- มีไฟล์ hex
- มีบอร์ดให้เลือก

ช่วยคิดและเก็งข้อสอบ midterm ให้หน่อย ว่าจะออกอะไร อ้างอิงจาก sources ทั้งหมดที่เคยอัปโหลดลง และมี reference book ที่ใช้ในการสอน คือหนังสือชื่อ The AVR Microcontroller and Embedded Systems Using Assembly and C; Using Arduino Uno and Atmel Studio, 2020 (SECOND EDITION: Based on ATMEGA328 and Arduino Boards, by MUHAMMAD ALI MAZIDI, et al.) ลองสมมติให้คุณคืออาจารย์วิชานี้ คือ Microprocessor and Interfacing วิชาของนักศึกษาปี 3 คณะ computer engineering ซึ่งแนวข้อสอบที่บอกคือมี 2 ข้อ ใช้เวลาสอบ 3 ชั่วโมง มีรายละเอียดคร่าวๆดังนี้
ข้อ 1 วัด Programming skills มั้ง ให้เขียน Assembly code(Assembly language) โจทย์ถามแนวแบบให้ implement the assembly code according to the algorithm that I'm asking you ถามเพื่อนมา ข้อสอบเก่าเคยออก ให้หา sorting ให้รันบน memory ram? assembly code or c ก็ได้(ไม่ต้องใช้ picsimlab) เขียนแค่ใน AVR Atmel Studio **ATmega328P**
ข้อ 2 เกี่ยวกับ How to connect your code(maybe from answer 1?) หรือไม่ก็เขียนมาใหม่มั้ง ไม่แน่ใจ แต่หลักๆคือข้อสอบจะให้ทำ How to connect your code with external devices (implement something by using emulator) at PICSimLab(emulator) and visualize the result.
ช่วยสอนเราแต่ละเรื่องที่เราควรรู้ที่จะช่วยให้ทำข้อสอบได้หน่อย

วิเคราะห์และเก็งข้อสอบ (Exam Prediction)

ข้อ 1: Programming Skills (Assembly Algorithm)

โจทย์: "จงเขียน Code Assembly เพื่อแก้ปัญหาตาม Algorithm ที่กำหนด" **แนวที่น่าจะออก:** การจัดการข้อมูลจำนวนมาก (Array/Block of data) หรือการคำนวณทางคณิตศาสตร์

- **เก็ง 1 (Data Block Manipulation):** โจทย์อาจสั่งให้ย้ายข้อมูลจาก Memory จุดหนึ่งไปยังอีกจุดหนึ่ง (เหมือน Lab 4-B Activity 2) หรือให้หาผลรวม (Summation) ของข้อมูลใน Array ที่เก็บอยู่ใน RAM หรือ Flash Memory

- **เก็ง 2 (Look-up Table & Calculation):** การคำนวณสมการคณิตศาสตร์ เช่น $Y = X^2 + 5$ โดย X เป็น Input และใช้ **Look-up Table** ในการหาค่า X^2 (เหมือน Lab 4-B Activity 3)

- **เก็ง 3 (16-bit Arithmetic):** การบวก/ลบเลขขนาด 16-bit หรือ BCD เพราะ Register เรามีแค่ 8-bit ต้องใช้ `ADD` ตามด้วย `ADC` (เหมือน Lab 3-D)

ข้อ 2: Interfacing & Simulation (PICSimLab)

โจทย์: "จงต่อวงจรและเขียน Code ควบคุมอุปกรณ์ภายนอก" **แนวที่น่าจะออก:** การใช้งาน I/O Ports ร่วมกับ Logic และ Timing

- **เก็ง 1 (7-Segment Display):** ฮิตที่สุดในแล็บ 4-C คือการแสดงตัวเลขบน 7-Segment อาจจะเป็นการนับเลข (Counter) หรือรับ Input จาก Switch แล้วเปลี่ยนตัวเลขที่แสดงผล

- **เก็ง 2 (LED Chaser/Pattern):** ไฟวิ่งไป-กลับ หรือติดดับตามเงื่อนไข Input (Switch) (เหมือน Lab 3-C)

- **จุดสำคัญ:** ต้องแม่นเรื่อง **Direction Register (DDRx)**, **Port Register (PORTx)** และ **Input Pin Register (PINx)**

Manual

2. วิธีทดสอบใน Atmel Studio (Simulation)

เพื่อเช็คว่าโค้ดทำงานถูกไหมโดยไม่ต้องต่อบอร์ดจริง:

1. **Build:** กด **F7** หรือเมนู **Build -> Build Solution** เพื่อสร้างไฟล์ `.hex` และเช็คว่าไม่มี Error [1]

2. **Start Debug:** กด **Alt+F5** หรือเมนู **Debug -> Start Debugging and Break** [2]

   - ถ้ามีหน้าต่างเด้งถาม ให้เลือก **Simulator** [2]

3. **เปิดหน้าต่างดูผล:**

   - ไปที่เมนู **Debug -> Windows -> I/O View** [3] -> เลือก **PORTD** (จะเห็นตารางบิตสี่เหลี่ยมๆ)

   - ไปที่เมนู **Debug -> Windows -> Processor Status** [3] -> ดูค่า **Register R20**

4. **รันทีละบรรทัด (Step):**

   - กด **F11 (Step Into)** หรือ **F10 (Step Over)** ย้ำๆ [4]

   - **สังเกตผล:**

     - ใน **Processor Status**: ค่า `R20` จะเพิ่มขึ้นทีละ 1 (0x00 -> 0x01 -> 0x02...)

     - ใน **I/O View**: ช่องสี่เหลี่ยมของ **PORTD** จะเปลี่ยนสี (ดำ/ขาว หรือ แดง/เทา) ตามค่า Binary ของตัวเลขนั้น

   - *เทคนิค:* ตอนถึงบรรทัด `RCALL DELAY` ให้กด **Shift+F11 (Step Out)** เพื่อข้ามการวนลูปรอ จะได้ไม่ต้องกด F11 เป็นพันครั้ง [5]

กด **Alt+F5** อีกรอบเพื่อรันโปรแกรมก่อน -> ไปที่เมนู Debug -> Windows -> Processor Status -> กดขยายตรงคำว่า **Registers**

---

3. วิธีทดสอบใน PICSimLab (Visualization)

เพื่อดูผลลัพธ์เสมือนจริง:

1. **เตรียมไฟล์:** ไปที่ Folder โปรเจกต์ของคุณ เข้าไปใน `Debug` จะเจอไฟล์นามสกุล **`.hex`**

2. **เปิด PICSimLab:**

   - เลือก Board: **Arduino Uno**

   - เลือก Microcontroller: **atmega328p**

3. **ต่ออุปกรณ์ (Spare Parts):**

   - เมนู **Modules -> Spare Parts**

   - เลือก **Outputs -> LEDs** [6]

   - ในหน้าต่าง LEDs ให้ติ๊กเลือกต่อสายไปที่ **PORT D** (Pin 0-7)

4. **โหลดโค้ด:**

   - เมนู **File -> Load Hex** [7] -> เลือกไฟล์ `.hex` ที่ได้จากข้อ 1

5. **ดูผลลัพธ์:**

   - คุณควรจะเห็นหลอดไฟ LED ติดเรียงกันเป็นเลขฐานสอง (Binary Count Up) วิ่งขึ้นเรื่อยๆ ครับ

วิธีการเลือกขา (Pin Selection) ให้ถูกต้อง Output: LEDs (Test Binary Counter) at Properties(คลิกขวา)

ในช่องที่คุณกดออกมา (Dropdown List) มันจะมีรายการยาวเหยียด ให้คุณเลื่อนหาบรรทัดที่เขียนว่า:

- สำหรับหลอดที่ 1: `0` หรือ `D0` หรือ `PD0` (มักจะอยู่บนๆ ของลิสต์)
- สำหรับหลอดที่ 2: `1` หรือ `D1` หรือ `PD1`
- สำหรับหลอดที่ 3: `2` หรือ `D2` หรือ `PD2`
- ... ไล่ไปเรื่อยๆ จนถึง ...
- สำหรับหลอดที่ 8: `7` หรือ `D7` หรือ `PD7`

ทำไมต้องเลือกเลขนี้? เพราะบนบอร์ด Arduino Uno ขา **PORTD** (ที่เราเขียนโค้ด `OUT PORTD`) มันตรงกับขา **Digital 0 ถึง 7** (D0-D7) บนบอร์ดครับ

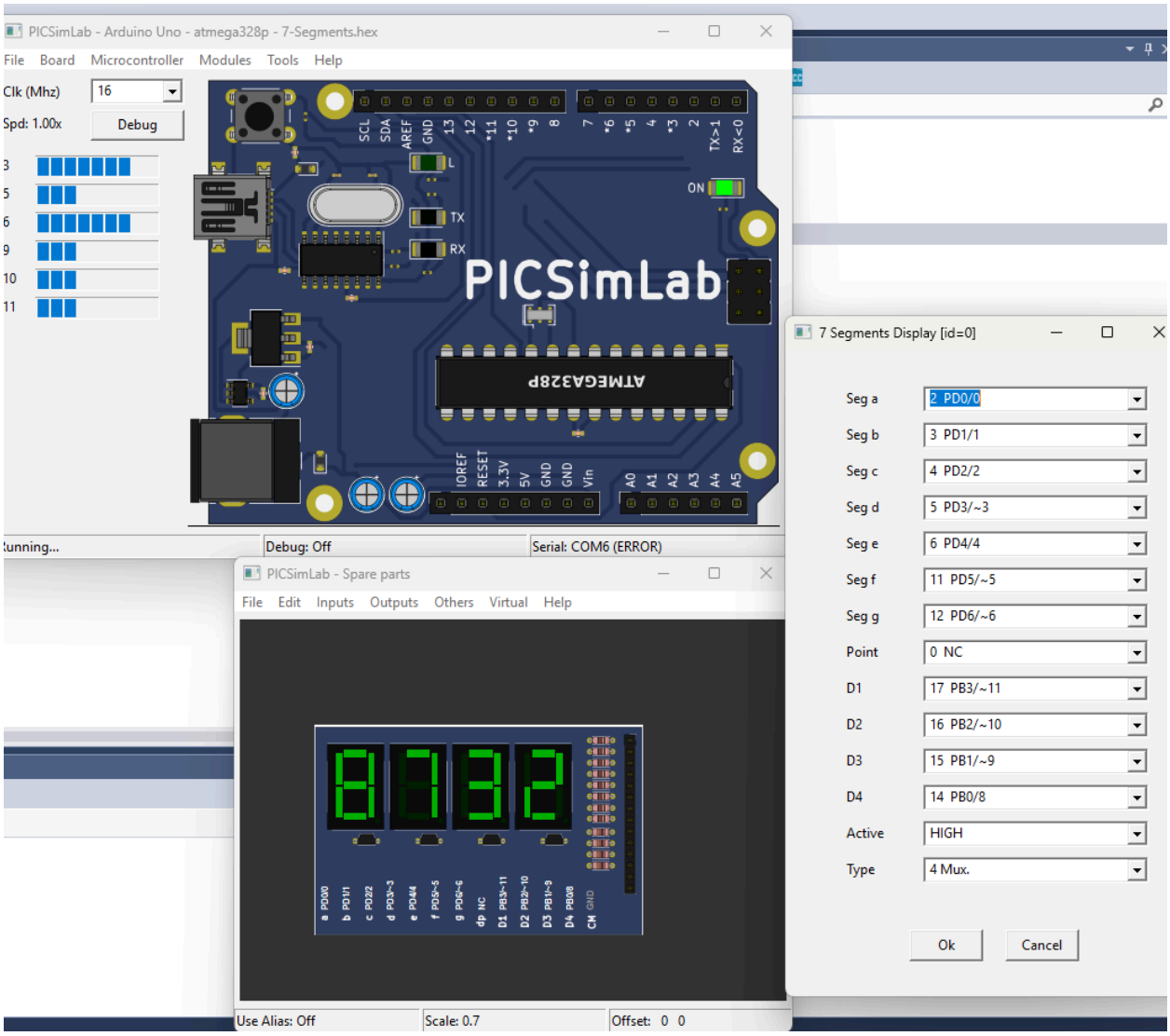
7-Segments (Lab 4-C)

Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	1	0	1	1	9

a เริ่มจากฝั่งขวา(LSB)

```
; --------------------
; 7-seg bit map (abcdefg, bit0=a ... bit6=g)  Common Cathode (1=ON)
; --------------------
DIG0    EQU 0b00111111'  ; a b c d e f
DIG1    EQU 0b00000110'  ; b c
DIG2    EQU 0b01011011'  ; a b d e g
DIG3    EQU 0b01001111'  ; a b c d g
DIG4    EQU 0b01100110'  ; b c f g
DIG5    EQU 0b01101101'  ; a c d f g
DIG6    EQU 0b01111101'  ; a c d e f g
DIG7    EQU 0b00000111'  ; a b c
```

```
DIG8    EQU 0b01111111'  ; a b c d e f g
DIG9    EQU 0b01101111'  ; a b c d f g
```



Board: Arduino Uno
Microcontroller: atmega328p
Output: 7-Segments
LDI R16, 0b01011011 (segment: a=1, b=1, c=0, … เรียงจากขวาไปซ้าย ให้เป็นเลข 2)
ถ้าสาย(D1-D4 เรียงจากซ้ายไปขวา)ไหนไม่ใช้ ให้ต่อ GND
ถ้าจะเพิ่มตำแหน่ง D(Digit) ให้ก็อปโค้ดเดิม และก็เปลี่ยนตรง D1-D4 เป็น PB4-PB0 (reverse)
@properties

GCC Application1
Output: LCD hd44780

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Ex: Dec 69 = 0x45 = Hex 45 = E

Basic Assembly

Instruction	Definition
ST	เอาไปใส่ลิ้นชัก(Address)
LD	หยิบจากลิ้นชัก(Address)
LDI	ใส่ค่า (Load Immediate)
INC	เพิ่มค่า (Increment)
DEC	ลดค่า (Decrement)
CP / CPI	เปรียบเทียบ (Compare/ Compare If)
LOOP	กระโดดกลับ/ วนซ้ำ
MOV	เปลี่ยนค่า

BRANCH	กระโดด
BREQ	กระโดดถ้า ค่าเท่ากับ… (Branch Equal)
BRNE	กระโดดถ้า ค่าไม่เท่ากับ… (Branch Not Equal)
BRLO	ถ้า**ตัวแรก** “น้อยกว่า” จะ…
BRSH	ถ้า**ตัวแรก** “เท่ากับหรือมากกว่า” จะ…
ADIW	เลื่อนนิ้ว(Pointer) ไปที่ค่าอีก Address นึง

Lecture 06: Page 24 Conditional Jump in AVR

```
LDI R16, 0

loop:
        INC  R16
        CPI  R16, 10
        BRNE loop
```

อ่านแบบนี้:
เริ่ม 0 เพิ่มค่า เช็คถึง 10 ยัง
ถ้ายัง → กลับไปเพิ่ม
จนสุดท้าย r16 = 10

1) MOV = ย้ายค่าระหว่าง REGISTER
        MOV R17, R16
        เอาค่าจาก R16 → ไปใส่ R17

2) ADD / SUB = บวก / ลบ
```
        ADD  R16, R17
        // R16 = R16 + R17
        SUB  R16, R17
        // R16 = R16 - R17
```

3) RJMP = กระโดดไป label
```
        RJMP  LOOP
```
        กระโดดไปที่ LOOP

4) LD / ST = อ่านเขียน RAM (🔥 สำคัญมากสำหรับข้อสอบ)
        อ่านจาก RAM (Load)

LD R16, X

เขียนลง RAM (Store)
ST X, R16

5) OUT = ส่งค่าไป PORT (ใช้ในข้อ 2)
OUT PORTB, R16
เอาค่าใน R16 → ไปออกขา

6) IN = อ่านจากอุปกรณ์
IN R16, PINB

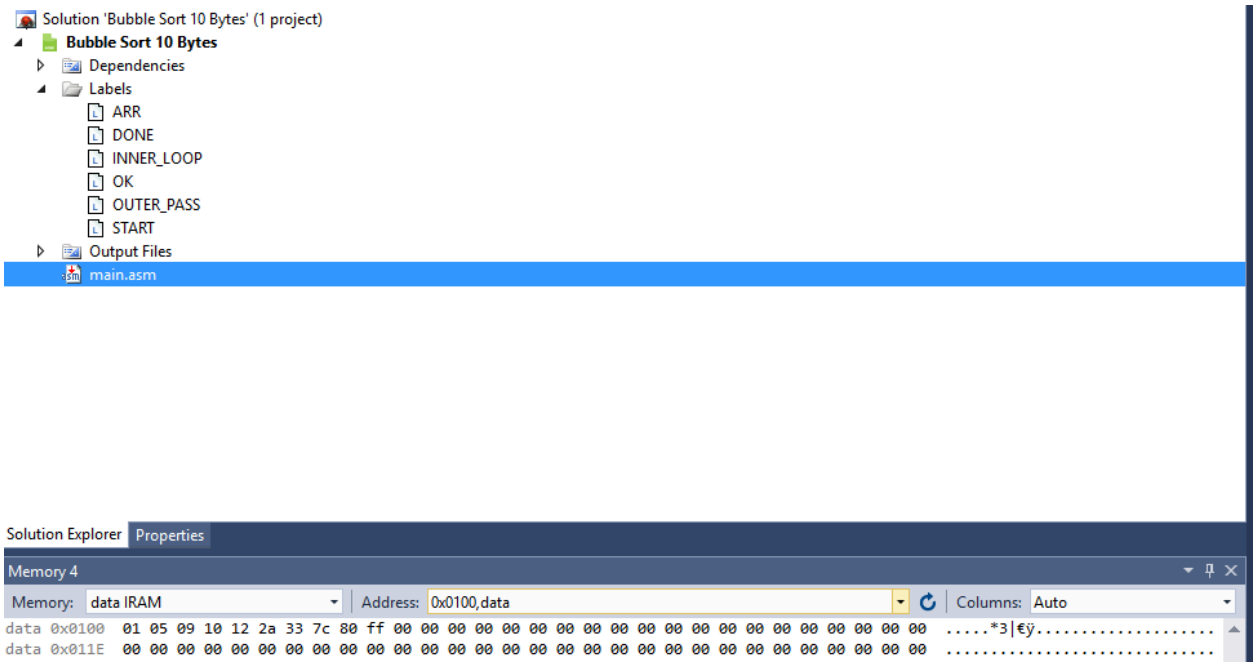
LDM #n	Immediate addressing. Load the number n to ACC
LDD <address>	Direct addressing. Load the contents of the location at the given address to ACC
LDI <address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC
LDX <address>	Indexed addressing. Form the address from <address> +the contents of the index register. Copy the contents of this calculated address to ACC
LDR #n	Immediate addressing. Load the number n to IX
STO <address>	Store the contents of ACC at the given address
ADD <address>	Add the contents of the given address to the ACC
INC <register>	Add 1 to the contents of the register (ACC or IX)
JMP <address>	Jump to the given address
CMP <address>	Compare the contents of ACC with the contents of <address>
CMP #n	Compare the contents of ACC with number n
JPE <address>	Following a compare instruction, jump to <address> if the compare was True
JPN <address>	Following a compare instruction, jump to <address> if the compare was False
END	Return control to the operating system

ให้หา keyword พวกนี้:

คำในโจทย์	แปลว่า block
transfer / move	copy
arrange	sort
mean	average
exceed	greater than
determine	find
total	sum

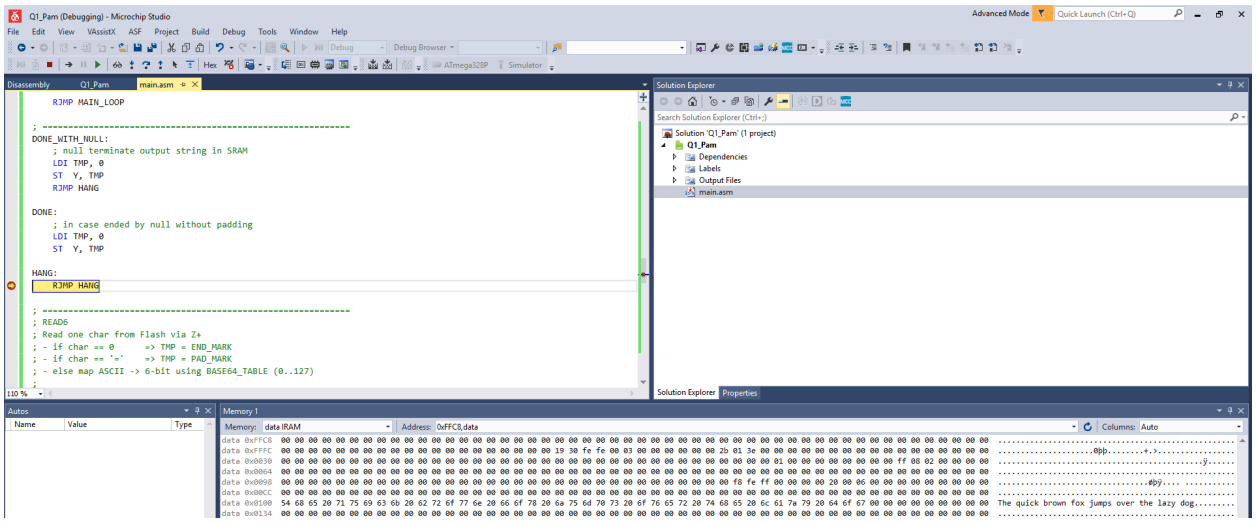
HEXADECIMAL VALUES																DECIMAL RANGE
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	00-15
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	16-31
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	32-47
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	48-63
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	64-79
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	80-95
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	96-111
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	112-127
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	128-143
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	144-159
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	160-175
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	176-191
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	192-207
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	208-223
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	224-239
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	240-255

Bubble Sort 10 bytes (Ascending, low to high)

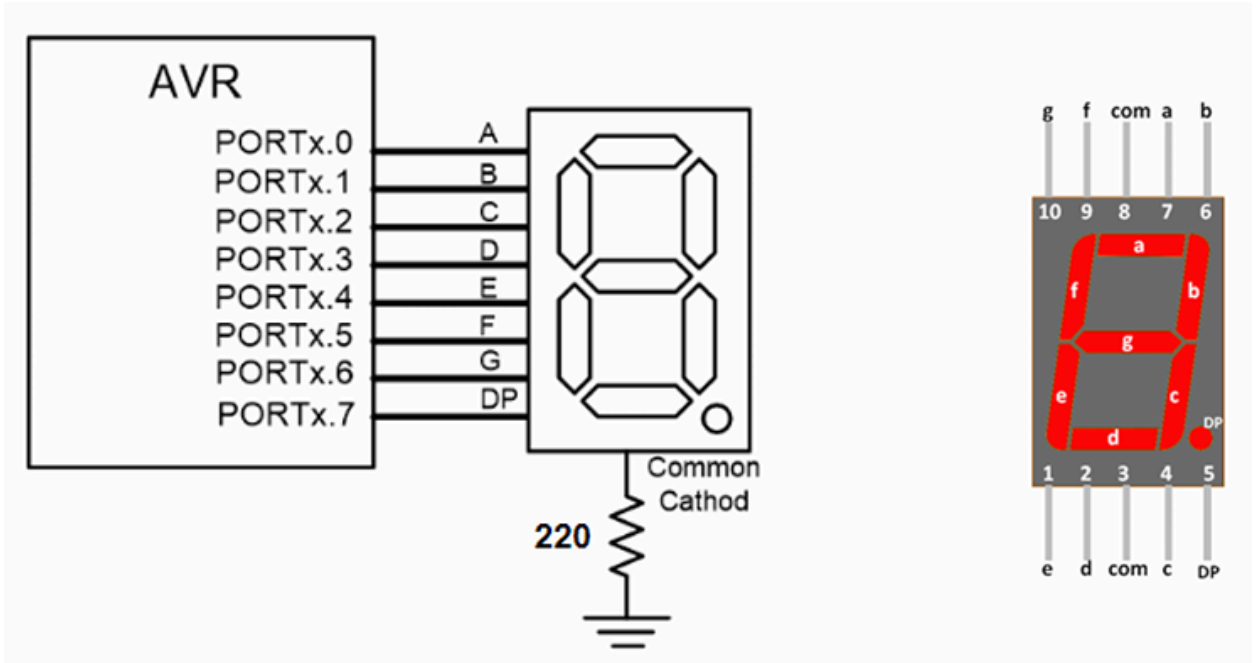


Sorting: อ่านค่า(debugging F5)ที่ Address ARR 0x100 (data IRAM)
Answer(Already Sorted): data 0x0100 01 05 09 10 12 2a 33 7c 80 ff

Q1_Senior (Base64 Decoder): insert breakpoint



Answer(Already decoded): data 0x0100 54 68 65 20 71 75 69 63 6b 20 62 72 6f 77 6e 20 66 6f 78 20 6a 75 6d 70 73 20 6f 76 65 72 20 74 68 65 20 6c 61 7a 79 20 64 6f 67 The quick brown fox jumps over the lazy dog



Q2_2025

2. Write the program to construct the "Bit logical operation" which includes Complement, And, Or, and Exclusive-Or operation. The program will receive 2 Bit inputs from switch 0 and 1 and show their status in LEDs 0 and 1 (Turn On = 1, Turn Off = 0), respectively. Switch 2 and 3 are used to define the logic operation as described in Table 1 and will be displayed in 7-segment as shown in Figure 1. Finally, the result of operation is displayed in LED 3.

Table 1: Switches status represent Logic Operation

Logic Operation	Switch 1	Switch 0
Not	0	0
Or	0	1
And	1	0
XOR	1	1

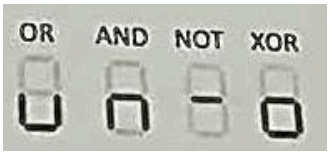


Figure 1: 7-Segment Logic Operation Display

Hint:

- Use PICSimLab to check your code
- Recommend to use “time delay code” between two outputs to make sure you get correct displays.