PRINCE OF SONGKLA UNIVERSITY
FACULTY OF ENIGNEERING

Final Examination : Semester II — Academic Year : 2005
Date : 27 February 2005 — Time: 13:30 – 16:30
Subject : 240- 305 Microprocessor and Assembly Lang. — Room : A201

คำสั่ง

1. ข้อสอบมีทั้งหมด 8 ข้อ (190 คะแนน)
2. เขียนคำตอบลงในสมุดคำตอบ

ทุจริตโทษต่ำสุดปรับตกวิชานี้ และพักการเรียน 1 ภาคการศึกษา โทษสูงสุดไล่ออก

	คะแนนที่ได้	คะแนนเต็ม
1		10
2		25
3		5
4		20
5		20
6		30
7		60
8		20
รวม		190

***** ห้ามนำเอกสารเข้าห้องสอบ *****

1. จงอธิบายความหมายของ *"write-through 8KB code and data cache set up as four-way set-associative"* ที่พบใน cache ของไมโครโพรเซสเซอร์ 80486 (10 คะแนน)

2. จากคำสั่งของ 80386 ที่ใช้ในการ set up โหมด protection ของการทำ segmentation พบว่ามีข้อมูลของ Descriptor จำนวน 8 ไบต์ดังในรูปที่ 1 จงอธิบายการทำงานของโปรแกรมและคุณลักษณะของ Segment ที่สร้างขึ้น

```
mov   ax, 0005h    ; load register AX with 0005h
lldt  ax           ;copy register AX to LDTR
mov   ax, 0017h    ; load register AX with 0017h
mov   cs, ax       ; copy register AX to segment register CS
```

(25 คะแนน)

3. จงตอบคำถามของไมโครคอนโทรลเลอร์ตระกูล 8051 ดังต่อไปนี้ (ช่องว่างละ 1 คะแนน)

   a. หนึ่งแมชชีนไซเคิลของ 8051 ใช้เวลา ____คาบเวลาออสซิลเลเตอร์
   b. คำสั่งที่ต้องการ 3 แมชชีนไซเคิลเมื่อใช้ Crystal 16 MHz จะใช้เวลา _____ไมโครวินาที
   c. 8051 สามารถอ่านข้อมูลหน่วยความจำได้สูงสุดไม่เกิน ____ Kbytes
   d. หน่วยความจำภายใน 8051 มีจำนวนทั้งหมด ____ Bytes
   e. รีจิสเตอร์ที่ทำหน้าที่บอกสภาวะการทำงานต่างๆ (Flag) คือ _____

(รวม 5 คะแนน)

4. จงออกแบบวงจรการเชื่อมต่อหน่วยความจำโปรแกรมภายนอกด้วย EPROM จำนวน 4 ตัว โดยแต่ละให้อยู่ในช่วง 0000 – 1FFF, 2000 – 3FFF, 4000 – 5FFF, 6000 – 7FFF

(20 คะแนน)

5. จงเขียนโปรแกรมภาษาแอสแซมบลี้สำหรับไมโครคอนโทรลเลอร์ตระกูล MCS51 ให้ LED ติดสว่างตามลูกเล่นแล้วแต่จะออกแบบ โดยให้ลอจิก '1' มีผลให้ LED สว่าง ในขณะที่ลอจิก '0' ทำให้ LED ดับ ซึ่งกำหนดให้ขับ LED ผ่านออกทาง port 2

(20 คะแนน)

6. จงเขียนโปรแกรมภาษาแอสแซมบลี้ รับข้อมูลผ่าน external input port ที่อยู่ ณ ตำแหน่ง 1AFH เก็บใส่ไว้ใน Accumulator รีจิสเตอร์ พร้อมกันนี้ให้วาดวงจรแสดงการขยาย port ในที่นี้สามารถใช้ไอซี 74LS244 เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอก

(30 คะแนน)

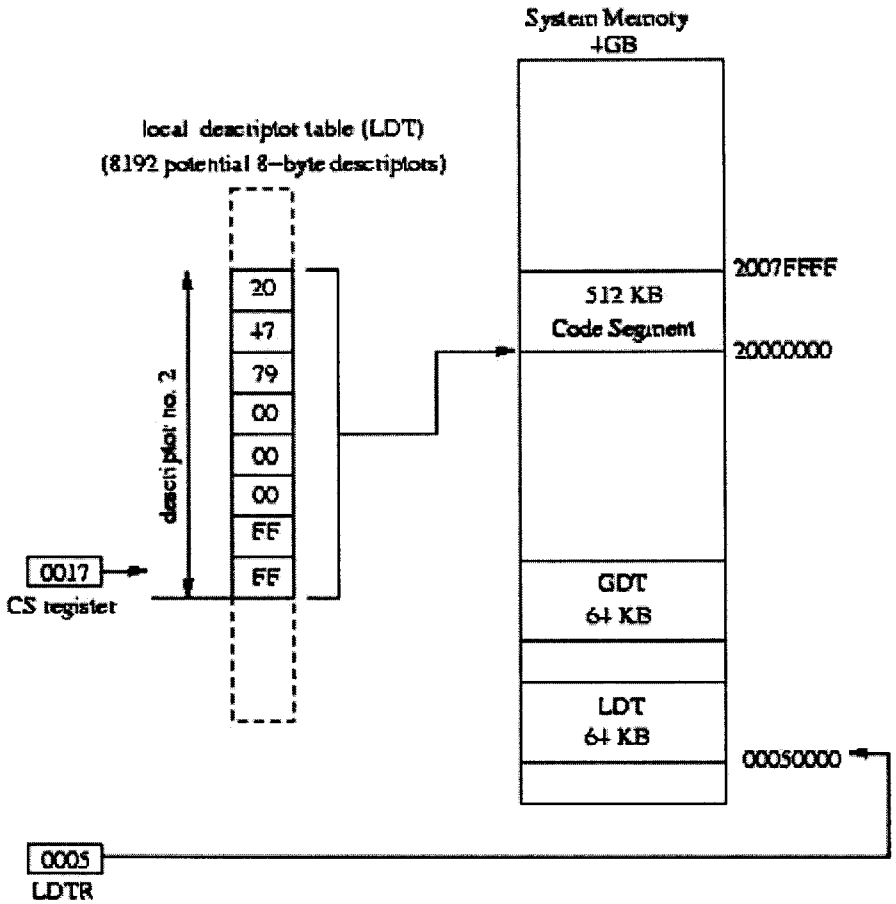
7. เขียนโปรแกรมภาษาแอสแซมบลี้การใช้งานอินเตอร์รัปต์แบบต่างๆ โดยเมื่อมีสัญญาณการอินเตอร์รัปต์ให้ทำการหมุนค่าใน accumulator ไปทางขวา แล้วเก็บค่าไว้ในหน่วยความจำตำแหน่ง 25h

   7.1 ใช้ External Interrupt 1 (20 คะแนน)

   7.2 ใช้ Timer1 โหมด 0 (20 คะแนน)

   7.2 ใช้ Timer0 โหมด 2 (20 คะแนน)

8. เขียนโปรแกรมภาษาแอสแซมบลี้หน่วงเวลาโดยใช้การอินเตอร์รัปต์ของวงจรจับเวลา ในที่นี้ต้องการให้หน่วงเวลา 1ms กำหนดให้ 8051 ใช้สัญญาณนาฬิกาที่ความถี่ 12 MHz (20 คะแนน)


System Memory
4GB
local descriptor table (LDT)
(8192 potential 8-byte descriptors)
descriptor no. 2
20
47
79
00
00
00
FF
FF
0017
CS register
2007FFFF
512 KB
Code Segment
20000000
GDT
64 KB
LDT
64 KB
00050000
0005
LDTR

รูปที่ 1

ตารางสัญลักษณ์ที่ใช้

สัญลักษณ์	ความหมาย
Rn	รีจิสเตอร์ภายใน R0 - R7
direct	รีจิสเตอร์ SFR และหน่วยความจำข้อมูลภายใน
@Ri	ค่าแอดเดรสหน่วยความจำภายใน อ้างแอดเดรส โดยอ้อมผ่าน รีจิสเตอร์ R0 หรือ R1
#data	ค่าคงที่ขนาด 8 บิต (ค่าจาก 0 - FF)
#data16	ค่าคงที่ขนาด 16 บิต (ค่าจาก 0 - FFFF)
addr11	ค่าตำแหน่งแอดเดรสจำนวน 11 บิต สำหรับคำสั่ง ACALL หรือ AJMP
addr16	ค่าตำแหน่งแอดเดรสจำนวน 16 บิต สำหรับคำสั่ง LCALL หรือ LJMP
rel	ค่าออฟเซ็ต (offset) หรือค่าบอกความสัมพัทธ์ (ค่าจาก -17 ถึง 18 ไบต์)
bit	ตำแหน่งบิตของหน่วยความจำภายในที่อ้างถึง ได้แบบบิต หรือ SFR
@DPTR	แอดเดรสของหน่วยความจำภายนอก หรือพอยน์เตอร์ โดยอ้อมผ่าน รีจิสเตอร์ DPTR

คำสั่งประมวลผลทางคณิตศาสตร์ (Arithematic Operations)

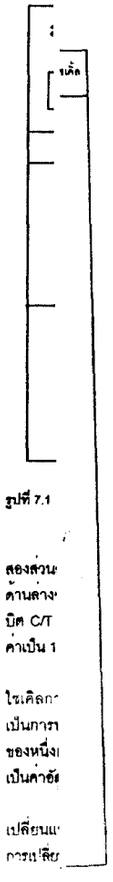
รูปแบบคำสั่ง		ความหมาย	จำนวนไบต์	จำนวนแมชชีนไซเคิล
ADD	A,Rn	บวกค่า Rn กับ A	1	1
ADD	A,direct	บวกค่าในหน่วยความจำ direct กับ A	2	1
ADD	A,@Ri	บวกค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A	1	1
ADDC	A,#data	บวกค่าคงที่ 8 บิตกับ A	2	1
ADDC	A,Rn	บวกค่า Rn กับ A พร้อมแฟลก Carry	1	1
ADDC	A,direct	บวกค่าในหน่วยความจำ direct กับ A พร้อมแฟลก Carry	2	1
ADDC	A,@Ri	บวกค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A พร้อมแฟลก Carry	1	1
ADDC	A,#data	บวกค่าคงที่ 8 บิตกับ A พร้อมแฟลก Carry	2	1
SUBB	A,Rn	ลบค่า Rn กับ A พร้อมแฟลก Borrow	1	1
SUBB	A,direct	ลบค่าในหน่วยความจำ direct กับ A พร้อมแฟลก Borrow	2	1
SUBB	A,@Ri	ลบค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A พร้อมแฟลก Borrow	1	1
SUBB	A,#data	ลบค่าคงที่ 8 บิตกับ A พร้อมแฟลก Borrow	2	1
INC	A	เพิ่มค่าใน A	1	1
INC	Rn	เพิ่มค่าใน Rn	1	1
INC	direct	เพิ่มค่าในหน่วยความจำ direct	2	1
INC	@Ri	เพิ่มค่าในหน่วยความจำที่เก็บอยู่ใน Ri	1	1
DEC	A	ลดค่าใน A	1	1
DEC	Rn	ลดค่าใน Rn	1	1
DEC	direct	ลดค่าในหน่วยความจำ direct	2	1
DEC	@Ri	ลดค่าในหน่วยความจำที่เก็บอยู่ใน Ri	1	1
INC	DPTR	เพิ่มค่าใน DPTR	1	2
MUL:	AB	คูณ A กับ B แล้วเก็บค่าใน A	1	4
DIV	AB	หาร A ด้วย B แล้วเก็บค่าใน A	1	4
DA	A	ทำ decimal adjust ค่าใน A	1	1

คำสั่งทางตรรก(Logical Operations)

รูปแบบคำสั่ง		ความหมาย	จำนวนไบต์	จำนวนแมชชีนไซเคิ้ล
ANL	A,Rn	AND ค่าใน Rn กับ A	1	1
ANL	A,direct	AND ค่าในหน่วยความจำ direct กับ A	2	1
ANL	A,@Ri	AND ค่าในหน่วยความจำที่เก็บใน Rn กับ A	1	1
ANL	A,#data	AND ค่าคงที่ 8 บิตกับ A	2	1
ANL	direct,A	AND ค่า A กับหน่วยความจำ direct	2	1
ANL	direct,#data	AND ค่าคงที่ 8 บิตกับหน่วยความจำ direct	3	2
ORL	A,Rn	OR ค่าใน Rn กับ A	1	1
ORL	A,direct	OR ค่าในหน่วยความจำ direct กับ A	2	1
ORL	A,@Ri	OR ค่าในหน่วยความจำที่เก็บอยู่ใน Rn กับ A	1	1
ORL	A,#data	OR ค่าคงที่ 8 บิตกับ A	2	1
ORL	direct,A	OR ค่า A กับหน่วยความจำ direct	2	1
ORL	direct,#data	OR ค่าคงที่ 8 บิตกับหน่วยความจำ direct	3	2
XRL	A,Rn	EX-OR ค่าใน Rn กับ A	1	1
XRL	A,direct	EX-OR ค่าในหน่วยความจำ direct กับ A	2	1
XRL	A,@Ri	EX-OR ค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A	1	1
XRL	A,#data	EX-OR ค่าคงที่ 8 บิตกับ A	2	1
XRL	direct,A	EX-OR ค่า A กับหน่วยความจำ direct	2	1
XRL	direct,#data	EX-OR ค่าคงที่ 8 บิตกับหน่วยความจำ direct	3	2
CLR	A	ทำค่าใน A ให้เป็นศูนย์	1	1
CPL	A	กลับค่าบิตใน A เป็นตรงข้ามทุกบิต	1	1
RL	A	หมุนบิตใน A ไปทางซ้าย 1 บิตและบิต 0 มีค่า 0	1	1
RLC	A	หมุนบิตใน A ไปทางซ้าย 1 บิต และบิต 0 เป็นค่าบิตที่อยู่ในแฟลก Carry	1	1
RR	A	หมุนบิตใน A ไปทางขวา 1 บิต และบิต 0 เป็นค่าจากบิต 7	1	1
RRC	A	หมุนบิตใน A ไปทางขวา 1 บิต และค่าจาก บิต 0 นำไปเก็บในแฟลก Carry และบิตที่อยู่ในแฟลก Carry เดิมจะย้ายมาเก็บในบิต 7	1	1
SWAP	A	สลับค่าสี่บิตบนกับสี่บิตล่างภายใน A	1	1

คำสั่งเคลื่อนย้ายข้อมูล (Data Transfer)

รูปแบบคำสั่ง		ความหมาย	จำนวนไบต์	จำนวนแมชชีนไซเคิ้ล
MOV	A,Rn	ย้ายข้อมูลจาก Rn ไป A	1	1
MOV	A,direct	ย้ายข้อมูลจากหน่วยความจำ direct ไป A	2	1
MOV	A,@Ri	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ใน Ri ไป A	1	1
MOV	A,#data	ย้ายค่าคงที่ 8 บิตไปเก็บที่ A	2	1
MOV	Rn,A	ย้ายข้อมูลจาก A ไป Rn	1	1
MOV	Rn,direct	ย้ายข้อมูลจากหน่วยความจำ direct ไป Rn	2	2
MOV	direct,A	ย้ายข้อมูลจาก A ไปยังหน่วยความจำ direct	2	1
MOV	direct,Rn	ย้ายข้อมูลจาก Rn ไปยังหน่วยความจำ direct	2	2
MOV	direct,direct	ย้ายข้อมูลระหว่างหน่วยความจำภายใน	3	2
MOV	direct,@Ri	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ใน Ri ไปยังหน่วยความจำ direct	2	2
MOV	direct,#data	ย้ายค่าคงที่ 8 บิตไปยังหน่วยความจำ direct	3	2
MOV	@Ri,A	ย้ายข้อมูลใน A ไปยังหน่วยความจำที่เก็บอยู่ใน Ri	1	1
MOV	@Ri,direct	ย้ายข้อมูลจากหน่วยความจำ direct ไปยังหน่วยความจำที่เก็บอยู่ใน Ri	2	2
MOV	@Ri,#data	ย้ายค่าคงที่ 8 บิตไปยังหน่วยความจำที่เก็บอยู่ใน Ri	2	1
MOV	DPTR,#data16	ย้ายค่าคงที่ 16 บิตไปยัง DPTR	3	2
MOVC	A,@A+DPTR	ย้ายข้อมูลจากหน่วยความจำข้อมูลที่สัมพัทธ์กับ DPTR ไปยัง A	1	2
MOVC	A,@A+PC	ย้ายข้อมูลจากหน่วยความจำข้อมูลที่สัมพัทธ์กับ PC ไปยัง A	1	2
MOVX	A,@Ri	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ใน Ri ไปยัง A	1	2
MOVX	A,@DPTR	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ใน DPTR ไปยัง A	1	2
MOVX	@Ri,A	ย้ายข้อมูลที่เก็บอยู่ใน A ไปยังหน่วยความจำที่เก็บอยู่ใน Ri	1	2
MOVX	@DPTR,A	ย้ายข้อมูลที่เก็บอยู่ใน A ไปยังหน่วยความจำที่เก็บอยู่ใน DPTR	1	2



รูปที่ 7.1

สองส่วน
ด้านล่าง
บิต C/T
ค่าเป็น 1

ไซเคิลก
เป็นการ
ของหนึ่ง
เป็นค่าอั

เปลี่ยนแ
การเปลี่ย

PUSH	direct	ย้ายข้อมูลหน่วยความจำ direct ไปเก็บยัง stack	2	2
POP	direct	ย้ายข้อมูลจาก stack ไปยังหน่วยความจำ direct	2	2
XCH	A,Rn	แลกเปลี่ยนข้อมูลระหว่าง A กับ Rn	1	1
XCH	A,direct	แลกเปลี่ยนข้อมูลระหว่างหน่วยความจำ direct กับ A	2	1
XCH	A,@Ri	แลกเปลี่ยนข้อมูลระหว่างหน่วยความจำที่เก็บอยู่ใน Ri กับ A	1	1
XCHD	A,@Ri	แลกเปลี่ยนข้อมูลสี่บิตล่างจากหน่วยความจำที่เก็บอยู่ใน Ri กับ A	1	1

คำสั่งจัดการข้อมูลแบบบิต (Boolean Variable Manipulation)

รูปแบบคำสั่ง		ความหมาย	จำนวนไบต์	จำนวนแมชชีนไซเคิล
CLR	C	ทำค่าแฟลก Carry ให้เป็น 0	1	1
CLR	bit	ทำค่า bit ให้เป็น 0	2	1
SETB	C	ทำค่าแฟลก Carry ให้เป็น 1	1	1
SETB	bit	ทำค่า bit ให้เป็น 1	2	1
CPL	C	กลับค่าแฟลก Carry ให้เป็นตรงข้าม	1	1
CPL	bit	กลับค่า bit ให้เป็นตรงข้าม	2	1
ANL	C,bit	AND ค่า bit กับแฟลก Carry	2	2
ANL	C,/bit	AND ค่าตรงข้ามของ bit กับแฟลก Carry	2	2
ORL	C,bit	ORL ค่า bit กับแฟลก Carry	2	2
ORL	C,/bit	ORL ค่าตรงข้ามของ bit กับแฟลก Carry	2	2
MOV	C,bit	ย้ายค่า bit มายังแฟลก Carry	2	1
MOV	bit,C	ย้ายค่าแฟลก Carry มายัง bit	2	2
JC	rel	กระโดด ถ้าค่าแฟลก Carry เป็น 1	2	2
JNC	rel	กระโดด ถ้าค่าแฟลก Carry เป็น 0	2	2
JB	bit,rel	กระโดด ถ้าค่า bit เป็น 1	3	2
JNB	bit,rel	กระโดด ถ้าค่า bit เป็น 0	3	2
JBC	bit,rel	กระโดด ถ้าค่า bit เป็น 1 และเปลี่ยนค่า bit เป็น 0	3	2

คำสั่งควบคุมการทำงานโปรแกรม (Program and Machine Control)

รูปแบบคำสั่ง		ความหมาย	จำนวนไบต์	จำนวนแมชชีนไซเคิล
CLR	C	ทำค่าแฟลก Carry ให้เป็น 0	1	1
ACALL	addr 11	ไปทำโปรแกรมย่อยจากค่าแอดเดรส 11 บิต	2	2
LCALL	addr 16	ไปทำโปรแกรมย่อยจากค่าแอดเดรส 16 บิต	3	2
RET		คำสั่งสิ้นสุดการทำงานโปรแกรมย่อย	1	2
RETI		คำสั่งสิ้นสุดการทำงานโปรแกรมย่อยอินเตอร์รัพต์	1	2
AJMP	addr 11	กระโดดไปยังตำแหน่งจากค่าแอดเดรส 11 บิต	2	2
LJMP	addr 16	กระโดดไปยังตำแหน่งจากค่าแอดเดรส 16 บิต	3	2
SJMP	rel	กระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน	2	2
JMP	@A+DPTR	กระโดดไปยังตำแหน่งที่สัมพัทธ์กับ DPTR	1	2
JZ	rel	กระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าหากค่า A เป็นค่า 0	2	2
JNZ	rel	กระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าหากค่า A เป็นค่า 1	2	2
CJNE	A,direct,rel	เปรียบเทียบ A กับหน่วยความจำ direct และกระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เท่ากัน	3	2
CJNE	A,#data,rel	เปรียบเทียบ A กับค่าคงที่ และกระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เท่ากัน	3	2
CJNE	Rn,#data,rel	เปรียบเทียบ Rn กับค่าคงที่ และ กระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เท่ากัน	3	2
CJNE	@Ri,#data,rel	เปรียบเทียบค่าในหน่วยความจำที่เก็บใน Ri กับค่าคงที่และกระโดดไปยังตำแหน่งที่ สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เท่ากัน	3	2
DJNZ	Rn,rel	ลดค่าใน Rn และกระโดดไปยังตำแหน่งที่ สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เป็น 0	2	2
DJNZ	direct,rel	ลดค่าในหน่วยความจำ direct และกระโดด ไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เป็น 0	3	2
NOP		ไม่มีการทำงานใดๆเกิดขึ้น	1	1

ชื่อบิต: TMOD	ตำแหน่ง: 89h			ค่าบิตเริ่มต้น: 0000 0000			
GATE1	C/T1	M1	M0	GATE0	C/T0	M1	M0

ชื่อบิต	ตำแหน่ง	ความหมาย
GATE1	TMOD.7	บิตควบคุม GATE สำหรับ Timer 1
C/T1	TMOD.6	บิตกำหนดการทำงานแบบตัวนับหรือจับเวลาของ Timer 1 โดยถ้าเป็นค่า 0 จะทำหน้าที่เป็นตัวจับเวลา
M1	TMOD.5	บิตบนสำหรับการกำหนดโหมดทำงานของ Timer 1
M0	TMOD.4	บิตล่างสำหรับการกำหนดโหมดทำงานของ Timer 1
GATE0	TMOD.3	บิตควบคุม GATE สำหรับ Timer 0
C/T0	TMOD.2	บิตกำหนดการทำงานแบบตัวนับหรือจับเวลาของ Timer 0 หากเป็นค่า 0 จะทำหน้าที่เป็นตัวจับเวลา
M1	TMOD.1	บิตบนสำหรับการกำหนดโหมดทำงานของ Timer 0
M0	TMOD.0	บิตล่างสำหรับการกำหนดโหมดทำงานของ Timer 0

รูปที่ 7.1 บิตต่างๆ ภายในรีจิสเตอร์ TMOD (Timer/Counter Mode Control)

ตามโครงสร้างของบิตภายในรีจิสเตอร์ TMOD ในรูปที่ 7.1 จะเห็นได้ว่ามีการจัดแบ่งออกเป็นสองส่วนอย่างชัดเจน โดยบิตจำนวนสี่บิตทางด้านบนจะเป็นของรีจิสเตอร์ T1 และสี่บิตที่เหลือทางด้านล่างจะเป็นของรีจิสเตอร์ T0 การกำหนดประเภทของการทำงานทำได้โดยการกำหนดค่าภายในบิต C/T ที่ตำแหน่งบิต 6 และ 2 โดยหากเป็นข้อมูลที่มีค่า 0 จะทำหน้าที่เป็นตัวจับเวลาและหากมีค่าเป็น 1 จะทำหน้าที่เป็นตัวนับสัญญาณ

เมื่อกำหนดให้ทำงานเป็นตัวจับเวลา รีจิสเตอร์จะทำการเพิ่มค่าขึ้นทีละหนึ่งในทุกๆ แมชชีนไซเคิลการทำงานของซีพียู ดังนั้นอาจจะกล่าวได้ในอีกลักษณะว่าการทำงานเป็นตัวจับเวลานี้เป็นการนับหน่วยเวลาซึ่งสร้างมาจากวงจรออสซิลเลเตอร์ของซีพียูเอง การคำนวณค่าระยะเวลาของหนึ่งแมชชีนไซเคิลจะใช้เวลานานเท่ากับคาบเวลาของออสซิลเลเตอร์จำนวน 12 คาบ หรือคิดเป็นค่าอัตราการนับในแต่ละครั้งจะใช้เวลาเท่ากับ 1/12 เท่าของความถี่ออสซิลเลเตอร์

กรณีกำหนดให้ทำงานเป็นการนับสัญญาณ รีจิสเตอร์จะเพิ่มค่าขึ้นทีละหนึ่งตามการเปลี่ยนแปลงสภาวะของสัญญาณทางขา T0 หรือ T1 การเปลี่ยนแปลงนี้จะเป็นในลักษณะของการเปลี่ยนจากระดับลอจิกสูงไปเป็นลอจิกต่ำ ซึ่งซีพียูจะทำการตรวจสอบสัญญาณนี้ทุกๆ แมชชีนไซเคิล

หากพบว่าแมชชีนไซเคิลแรกเป็นระดับลอจิกสูงและแมชชีนไซเคิลต่อไปเป็นระดับลอจิกต่ำ จึงทำการเพิ่มค่าในรีจิสเตอร์ ดังนั้นซีพียูจะต้องใช้เวลาถึง 2 แมชชีนไซเคิลในการตรวจสอบสัญญาณ ซึ่งเป็นผลทำให้อัตราการเปลี่ยนแปลงของสัญญาณอินพุตภายนอกทางขา T0 หรือ T1 มีได้สูงไม่เกินค่า 1/24 เท่าของความถี่ออสซิลเลเตอร์

7.3 การอินเตอร์รัปต์ของวงจรนับ/จับเวลา

จากกระบวนการทำงานของวงจรนับ/จับเวลาของ 8051 จำเป็นต้องทำการกำหนดค่าเริ่มให้กับรีจิสเตอร์ T0 หรือ T1 ค่านี้จะเป็นค่าจำนวนของพัลส์ภายในที่ต้องการจะให้นับหรือค่าจำนวนพัลส์ภายนอกที่เข้ามาทางขาสัญญาณ T0 และ T1 ค่าตัวเลขภายในรีจิสเตอร์นี้จะต้องให้มีค่าน้อยกว่าค่าที่ต้องการอยู่หนึ่งค่า ทั้งนี้เนื่องจากการทำงานของรีจิสเตอร์จะเพิ่มค่าจากที่กำหนดไปเรื่อยๆ จนถึงค่าสูงสุดของรีจิสเตอร์ และกลับไปเป็นค่าศูนย์เมื่อมีพัลส์สุดท้ายเกิดขึ้น ซึ่งเรียกว่ามี การโอเวอร์โฟล้ว เกิดขึ้น ทำให้เกิดการกำหนดค่าของแฟล็กเพื่อแจ้งให้ซีพียูรับทราบ ดังนั้นโปรแกรมทั่วไปจึงมักจะใช้สภาวะของแฟล็กนี้ (TF0 และ TF1) ซึ่งเป็นบิตอยู่ภายในรีจิสเตอร์ TCON (ดูรูปที่ 7.2) เพื่อตรวจสอบว่ากระบวนการนับได้เสร็จสิ้นลงแล้ว หรือใช้เพื่อทำอินเตอร์รัปต์โปรแกรมต่อไป

ชื่อบิต: TCON	ตำแหน่ง: 88h			ค่าบิตเริ่มต้น: 0000 0000			
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

ชื่อบิต	ตำแหน่ง	ความหมาย
TF1	TCON.7	แฟลกแสดงการอินเตอร์รัพต์ของ Timer 1
TR1	TCON.6	บิตเลือกประเภทสัญญาณอินเตอร์รัพต์ Timer 1
TF0	TCON.5	แฟลกแสดงการอินเตอร์รัพต์ของ Timer 0
TR0	TCON.4	บิตเลือกประเภทสัญญาณอินเตอร์รัพต์ Timer 0
IE1	TCON.3	แฟลกแสดงการอินเตอร์รัพต์ของ INT1
IT1	TCON.2	บิตเลือกประเภทสัญญาณอินเตอร์รัพต์ INT1
IE0	TCON.1	แฟลกแสดงการอินเตอร์รัพต์ของ INT0
IT0	TCON.0	บิตเลือกประเภทสัญญาณอินเตอร์รัพต์ INT0

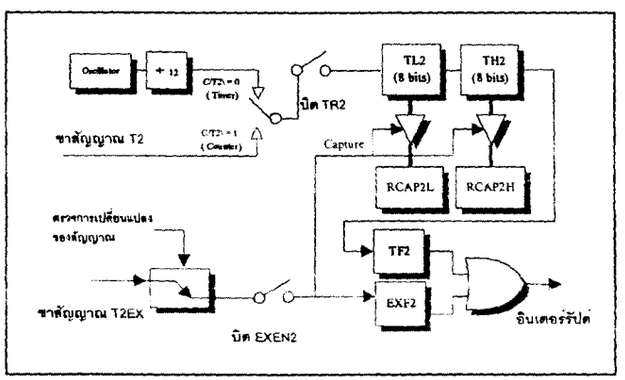
รูปที่ 7.2 บิตต่างๆ ภายในรีจิสเตอร์ TCON (Timer/Counter Control)

ชื่อบิต: T2CON ตำแหน่ง: C8h ค่าบิตเริ่มต้น: 0000 0000

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

ชื่อบิต	ตำแหน่ง	ความหมาย
TF2	T2CON.7	แฟลกโอเวอร์โฟล์วของ timer 2
EXF2	T2CON.6	Timer 2 External flag
RCLK	T2CON.5	Receive clock flag
TCLK	T2CON.4	Transmit clock flag
EXEN2	T2CON.3	Timer 2 external enable flag
TR2	T2CON.2	แฟลกกำหนดการทำงาน/หยุดการทำงานของ timer 2
C/T2	T2CON.1	แฟลกเลือกโหมดการทำงานเป็นตัวนับหรือจับเวลา
CP/RL2	T2CON.0	แฟลกแสดงโหมดการ Capture/Reload

รูปที่ 7.12 บิตต่างๆ ภายในรีจิสเตอร์ T2CON (Timer/Counter 2 Control Register)



รูปที่ 7.13 แผนภาพแสดงการทำงานในโหมดแคปเจอร์ของ Timer 2

Timer 2 จะทำหน้าที่เป็นตัวนับหรือตัวจับเวลา ซึ่งเมื่อเกิดการโอเวอร์โฟล์วขึ้นจะมี ทำให้แฟล็กสถานะ TF2 มีค่าเป็น 1 และสามารถนำไปสร้างการอินเตอร์รัปต์ขึ้นได้

2. เมื่อกำหนดค่าบิต EXEN2 เป็น 1

Timer 2 ยังคงทำงานเช่นเดียวกับลักษณะข้างต้นและมีคุณสมบัติเพิ่มขึ้น กล่าวคือ เ การเปลี่ยนแปลงระดับสัญญาณทางขาสัญญาณ T2EX จากลอจิกสูงเป็นลอจิกต่ำ จะมีผลทำใ ข้อมูลภายในรีจิสเตอร์ของ Timer 2 คือ TL2 และ TH2 ถูกนำไปใส่ (Capture) ให้กับรีจิสเตอร์ RCA และ RCAP2H ซึ่งเป็นรีจิสเตอร์หน้าที่พิเศษหรือ SFR ที่มีเฉพาะกับไมโครคอนโทรลเลอร์เบอร์ 8 เท่านั้น นอกจากนี้ยังจะมีผลทำให้บิต EXF2 ภายในรีจิสเตอร์ T2CON มีค่าเป็น 1 ขึ้นด้วย สามารถนำไปสร้างการอินเตอร์รัปต์ให้เกิดขึ้นได้เช่นกัน

7.6.2 โหมดการโหลดค่าโดยอัตโนมัติ

ค่าของบิต EXEN2	ผลการทำงาน
0	เมื่อเกิดการโอเวอร์โฟล์วของ Timer 2 นอกจากจะมีผลทำให้แฟล็ มีค่าเป็น 1 แล้ว ยังทำให้มีการนำค่าข้อมูลจากรีจิสเตอร์ RCAP2 RCAP2H มาใส่ให้กับรีจิสเตอร์ Timer 2 โดยอัตโนมัติ
1	Timer 2 ยังคงทำงานเหมือนกับลักษณะข้างต้น และมีคุณสมบัติ ดังนี้ เมื่อมีการเปลี่ยนแปลงระดับสัญญาณจากลอจิกสูงเป็นลอ ทางขาสัญญาณ T2EX ก็จะมีผลทำให้มีการนำค่าข้อมูลจากรีจิ RCAP2L และ RCAP2H มาใส่ให้กับรีจิสเตอร์ Timer 2 โดยอัต เช่นเดียวกัน และยังทำให้แฟล็ก TF2 มีค่าเป็น 1 ด้วย

โหมดการโหลดค่าโดยอัตโนมัติ (Auto-reload mode) นี้ สามารถเลือกทำงานได้สอ เช่นเดียวกัน โดยการกำหนดค่าให้กับบิต EXEN2 ภายในรีจิสเตอร์ T2CON (ดูรูปที่ 7.1

7.6.3 โหมดการกำเนิดอัตราบอด

สำหรับโหมดการกำเนิดอัตราบอด (Baud rate Generator) ของ Timer 2 จะมีความ จาก Timer 0 และ Timer 1 โดยวงจรด้านการรับและการส่งสามารถเป็นค่าที่ต่างกันได้ การกำหนดค่าให้กับบิต TCLK และ RCLK ของ Timer 2 ดังแสดงให้เห็นจากแผนภาพ ของโหมดนี้ในรูปที่ 7.15