

มหาวิทยาลัยสงขลานครินทร์
คณะวิศวกรรมศาสตร์



สอบกลางภาค: ภาคการศึกษาที่ 1

ปีการศึกษา: 2552

วันที่สอบ: 4 ตุลาคม 2552

เวลาสอบ: 1330-1630

รหัสวิชา: 241-307

ห้องสอบ: วิทยาลัย

ชื่อวิชา: COMPUTER SYSTEMS ARCHITECTURES AND ORGANIZATIONS

คำสั่ง

อ่านรายละเอียดของข้อสอบและคำแนะนำให้เข้าใจก่อนเริ่มทำข้อสอบ

อนุญาต : เครื่องเขียนต่างๆ ปากกา หรือดินสอ
: กระดาษโน้ตขนาด A4 จำนวน 1 แผ่น
: เครื่องคิดเลข

ไม่อนุญาต : Computer Notebook, หนังสือ และสมุดต่างๆ

เวลา: 3 ชั่วโมง (180 นาที)

คำแนะนำ

- ข้อสอบมีจำนวน 16 หน้า(รวมทั้งใบปะหน้าและภาคผนวก) ให้ทำทุกข้อ
- เขียนคำตอบลงในข้อสอบเท่านั้น
- อนุญาตให้ใช้ดินสอในการทำข้อสอบได้ กรณีเขียนไม่ชัดหรืออ่านไม่ออก จะถือว่าคำตอบนั้นผิด
- อ่านคำสั่งในแต่ละข้อให้เข้าใจก่อนลงมือทำ
- ให้เขียนชื่อ-นามสกุลและรหัสนักศึกษาในข้อสอบทุกแผ่น แผ่นใดไม่เขียนหรือเขียนไม่ครบจะถูกตัดคะแนนแผ่นละ 1 คะแนน
- อนุญาตให้ทศเลขลงด้านหลังของข้อสอบได้

-ทุจริตในการสอบมีโทษขั้นต่ำปรับตกในรายวิชานี้และพักการเรียน 1 ภาคการศึกษา-

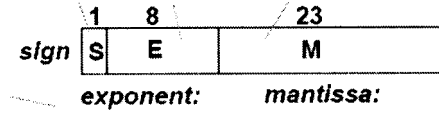
2. กำหนด A และ B เป็น Floating point แบบ IEEE754 single precision

โดย

A = 0100 0110 0101 1000 0000 0000 0000 0000

B = 0100 0001 1010 0111 0000 0000 0000 0000

C = A × B



จงหาว่า A และ B มีค่าในเลขฐานสิบเท่ากับเท่าใดและแสดงวิธีการคำนวณหาค่า C

แปลง A ให้เป็นเลขฐานสิบ (2 คะแนน)

.....

.....

.....

.....

.....

แปลง B ให้เป็นเลขฐานสิบ (2 คะแนน)

.....

.....

.....

.....

.....

แสดงวิธีการคำนวณหาค่า C (4 คะแนน)

.....

.....

.....

.....

.....

.....

.....

.....

วงจรสร้างสัญญาณ MDRout (6 คะแนน)

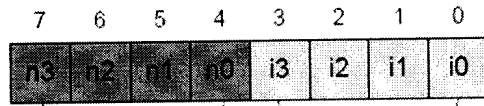
ชื่อ-สกุล.....รหัส.....

วงจรร่างสัญญาณ PCout (6 คะแนน)

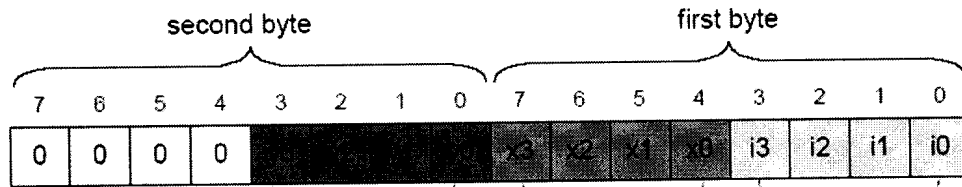
ภาคผนวก 1- ชุดคำสั่งของ Simple1

คำสั่ง	ความหมาย	ความยาวของคำสั่ง	ผลกระทบต่อแฟลก				
			C	S	Z	O	P
			F	F	F	V	F
INV Rn	กลับค่าบิตของรีจิสเตอร์ Rn เป็นตรงกันข้าม	1 byte			x		x
SETF b,v	เซตค่าบิตของแฟลกที่ระบุให้มีค่าตามที่กำหนด เช่น SETF CF,1	1 byte	?	?	?	?	?
Load Rn,[R1R0]	ก๊อปปี้ค่าในหน่วยความจำตำแหน่งที่ระบุมาใส่ในรีจิสเตอร์ Rn	1 byte					
Store [R1R0], Rn	ก๊อปปี้ค่าในรีจิสเตอร์ Rn มาใส่ในหน่วยความจำตำแหน่งที่ระบุ	1 byte					
Branch [R1R0]	บรานซ์แบบไม่มีเงื่อนไขไปยังแอดเดรสที่ระบุ	1 byte					
Branch ZF, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก ZF=1	1 byte					
Branch SF, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก SF=1	1 byte					
Branch PF, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก PF=1	1 byte					
Branch OV, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก OV=1	1 byte					
Branch CF, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก CF=1	1 byte					
ADD Rx,Ry	$Rx \leftarrow Rx + Ry$ (Addition)	2 bytes	x	x	x	x	x
ADDC Rx,Ry	$Rx \leftarrow Rx + Ry + CF$ (Addition with carry flag)	2 bytes	x	x	x	x	x
SUBB Rx,Ry	$Rx \leftarrow Rx - Ry - CF$ (Subtraction with borrow flag)	2 bytes	x	x	x	x	x
AND Rx,Ry	$Rx \leftarrow Rx \& Ry$	2 bytes			x		x
OR Rx,Ry	$Rx \leftarrow Rx Ry$	2 bytes			x		x
Load Rx,#data8	อ่านค่าคงที่ค่า #data8 มาใส่ในรีจิสเตอร์ Rx	2 bytes					

ภาคผนวก 2-ฟอร์มแมตของคำสั่งของ Simple1



- | | |
|----------|------------------------|
| Rn | 0000 = Branch |
| 0000=R0 | 0001 = INV |
| 0001=R1 | 0010 = SETF |
| 0010=R2 | 0011 = LOAD Rn,[R1,R0] |
| ... | 0100 = STORE |
| ... | 0101 = Branch PF |
| ... | 0110 = Branch SF |
| 1111=R15 | 0111 = Branch CF |
| | 1000 = Branch OV |
| | 1001 = Branch ZF |



- | | | |
|----------|----------|-----------------------|
| Ry | Rx | 1010 = ADD |
| 0000=R0 | 0000=R0 | 1011 = ADDC |
| 0001=R1 | 0001=R1 | 1100 = SUBB |
| 0010=R2 | 0010=R2 | 1101 = AND |
| ... | ... | 1110 = OR |
| ... | ... | 1111 = LOAD Rx,#data8 |
| 1111=R15 | 1111=R15 | |

ภาคผนวก 3- ตัวอย่าง instruction cycle ของ Simple1

Instruction Fetch Cycle ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แยกตัวในแต่ละคล็อกไซเคิล
0	MAR \leftarrow PC	PC _{out} , MAR _{in}
1	address_bus \leftarrow MAR, mem_read, Z \leftarrow PC+1, MDR \leftarrow mem[MAR]	PC _{out} , M _{read} , MDR _{inE} , Z _{in} , ALU _{add} , MUXselY=0
2	PC \leftarrow Z	PC _{in} , Z _{out}
3	IR \leftarrow MDR	IR _{in} , MDR _{out}

ไซเคิลการเอกซิดิวต์ของคำสั่ง ADDC Rx, Ry ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แยกตัวในแต่ละคล็อกไซเคิล
4	MAR \leftarrow PC	PC _{out} , MAR _{in}
5	address_bus \leftarrow MAR, mem_read, Z \leftarrow PC+1, MDR \leftarrow mem[MAR]	PC _{out} , M _{read} , MDR _{inE} , Z _{in} , ALU _{add} , MUXselY=0
6	PC \leftarrow Z	PC _{in} , Z _{out}
7	IR \leftarrow MDR	IR _H _{in} , MDR _{out}
8	Y \leftarrow Ry	Y _{in} , Ry _{out}
9	Z \leftarrow Rx + Y + CF	Rx _{out} , MUXselY=1, ALU _{ADDC} , Z _{in} , CF _{in} , OV _{in} , ZF _{in} , SF _{in} , PF _{in}
10	Rx \leftarrow Z	Z _{out} , Rx _{in} , End

ไซเคิลการเอกซิทวิตของคำสั่ง Load Rx,#data8 ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แยกตัวในแต่ละคล็อกไซเคิล
4	MAR \leq PC	PC _{out} , MAR _{in}
5	address_bus \leq MAR, mem_read, Z \leq PC+1, MDR \leq mem[MAR]	PC _{out} , M _{read} , MDR _{inE} , Z _{in} , ALU _{add} , MUXselY=0
6	PC \leq Z	PC _{in} , Z _{out}
7	Rx \leq MDR	Rx _{in} , MDR _{out} , End

ไซเคิลการเอกซิทวิตของคำสั่ง STORE [R1R0],Rn ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แยกตัวในแต่ละคล็อกไซเคิล
4	MAR _H \leq R1, MAR _L \leq R0	MAR _{in} , R1 _{out_to_H} , R0 _{out_int_bus}
5	MDR \leq Rn,	MDR _{in} , Rn _{out}
6	mem[MAR] \leq MDR, mem_write	MDR _{outE} , Mwrite
7	deactivate write signal	MDR _{outE} , End

ไซเคิลการเอกซิทวิตของคำสั่ง Branch [R1R0] ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แยกตัวในแต่ละคล็อกไซเคิล
4	PC _H \leq R1, PC _L \leq R0	PC _{in} , R1 _{out_to_H} , R0 _{out_int_bus} , End

ไซเคิลการเอกซิทวิตของคำสั่ง Branch OV,[R1R0] ของ Simple1

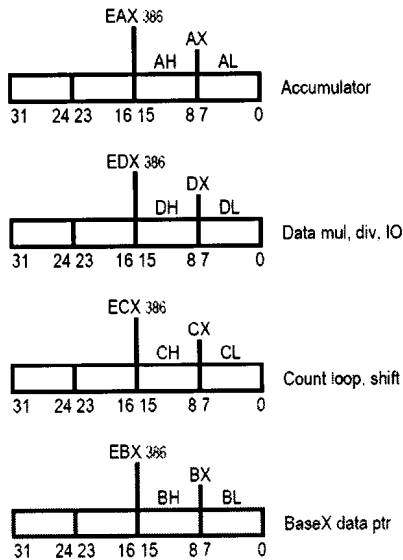
คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แยกตัวในแต่ละคล็อกไซเคิล
4	if OV=0 then End	End_if_flag_not_set
5	PC _H \leq R1, PC _L \leq R0	PC _{in} , R1 _{out_to_H} , R0 _{out_int_bus} , End

ภาคผนวก 4- 80x86 Instruction Set table

JUMPS (flags remain unchanged)							
Name	Comment	Code	Operation	Name	Comment	Code	Operation
CALL	Call subroutine	CALL Proc		RET	Return from subroutine	RET	
JMP	Jump	JMP Dest					
JE	Jump if Equal	JE Dest	(= JZ)	JNE	Jump if not Equal	JNE Dest	(= JNZ)
JZ	Jump if Zero	JZ Dest	(= JE)	JNZ	Jump if not Zero	JNZ Dest	(= JNE)
JCXZ	Jump if CX Zero	JCXZ Dest		JECXZ	Jump if ECX Zero	JECXZ Dest	386
JP	Jump if Parity (Parity Even)	JP Dest	(= JPE)	JNP	Jump if no Parity (Parity Odd)	JNP Dest	(= JPO)
JPE	Jump if Parity Even	JPE Dest	(= JP)	JPO	Jump if Parity Odd	JPO Dest	(= JNP)

JUMPS Unsigned (Cardinal)				JUMPS Signed (Integer)			
JA	Jump if Above	JA Dest	(= JNBE)	JG	Jump if Greater	JG Dest	(= JNLE)
JAE	Jump if Above or Equal	JAE Dest	(= JNB = JNC)	JGE	Jump if Greater or Equal	JGE Dest	(= JNL)
JB	Jump if Below	JB Dest	(= JNAE = JC)	JL	Jump if Less	JL Dest	(= JNGE)
JBE	Jump if Below or Equal	JBE Dest	(= JNA)	JLE	Jump if Less or Equal	JLE Dest	(= JNG)
JNA	Jump if not Above	JNA Dest	(= JBE)	JNG	Jump if not Greater	JNG Dest	(= JLE)
JNAE	Jump if not Above or Equal	JNAE Dest	(= JB = JC)	JNGE	Jump if not Greater or Equal	JNGE Dest	(= JL)
JNB	Jump if not Below	JNB Dest	(= JAE = JNC)	JNL	Jump if not Less	JNL Dest	(= JGE)
JNBE	Jump if not Below or Equal	JNBE Dest	(= JA)	JNLE	Jump if not Less or Equal	JNLE Dest	(= JG)
JC	Jump if Carry	JC Dest		JO	Jump if Overflow	JO Dest	
JNC	Jump if no Carry	JNC Dest		JNO	Jump if no Overflow	JNO Dest	
				JS	Jump if Sign (= negative)	JS Dest	
				JNS	Jump if no Sign (= positive)	JNS Dest	

General Registers:



Example:

```

.DOSSEG ; Demo program
.MODEL SMALL
.STACK 1024
Two EQU 2 ; Const
.DATA
VarB DB ? ; define Byte, any value
VarW DW 1010b ; define Word, binary
VarW2 DW 257 ; define Word, decimal
VarD DD 0AFFFh ; define Doubleword, hex
S DB "Hello !" ; define String
.CODE
main: MOV AX,DGROUP ; resolved by linker
MOV DS,AX ; init datasegment reg
MOV [VarB],42 ; init VarB
MOV [VarD],-7 ; set VarD
MOV BX,Offset[S] ; addr of "H" of "Hello !"
MOV AX,[VarW] ; get value into accumulator
ADD AX,[VarW2] ; add VarW2 to AX
MOV [VarW2],AX ; store AX in VarW2
MOV AX,4C00h ; back to system
INT 21h
END main
    
```

Flags:

Control Flags (how instructions are carried out):
D: Direction 1 = string op's process down from high to low address
I: Interrupt whether interrupts can occur. 1= enabled
T: Trap single step for debugging

Status Flags (result of operations):

C: Carry result of unsigned op. is too large or below zero. 1 = carry/borrow
O: Overflow result of signed op. is too large or small. 1 = overflow/underflow
S: Sign sign of result. Reasonable for Integer only. 1 = neg. / 0 = pos.
Z: Zero result of operation is zero. 1 = zero
A: Aux. carry similar to Carry but restricted to the low nibble only
P: Parity 1 = result has even number of set bits