ชื่อ-สกุล.....รหัส.....

มหาวิทยาลัยสงขลานครินทร์

คณะวิศวกรรมศาสตร์

สอบกลางภาค: ภาคการศึกษาที่ 1 **ปีการศึกษา:** 2552

วันที่สอบ: 2 ตุลาคม 2552 **เวลาสอบ:** 09.00-12.00

รหัสวิชา: 240-208 **ห้องสอบ:** R300

ชื่อวิชา: Fundamentals of Computer Architecture

คำสั่ง

อ่านรายละเอียดของข้อสอบและคำแนะนำให้เข้าใจก่อนเริ่มทำข้อสอบ

อนุญาต : เครื่องเขียนต่างๆ ปากกา หรือดินสอ

: กระดาษโน๊ตขนาด A4 จำนวน 1 แผ่น

: เครื่องคิดเลข

ไม่อนุญาต : Computer Notebook, หนังสือ และสมุดต่างๆ

เวลา: 3 ชั่วโมง (180 นาที)

คำแนะนำ

- ข้อสอบมีจำนวน 15หน้า(รวมทั้งใบปะหน้าและภาคผนวก) ให้ทำทุกข้อ
- เขียนคำตอบลงในข้อสอบเท่านั้น
- อนุญาตให้ใช้ดินสอในการทำข้อสอบได้ กรณีเขียนไม่ชัดหรืออ่านไม่ออก จะถือว่าคำตอบนั้นผิด
- อ่านคำสั่งในแต่ละข้อให้เข้าใจก่อนลงมือทำ
- ให้เขียนชื่อ-นามสกุลและรหัสนักศึกษาในข้อสอบทุกแผ่น แผ่นใดไม่เขียนหรือเขียนไม่ครบจะถูกตัดคะแนนแผ่นละ 1 คะแนน
- อนุญาตให้ทดเลขลงด้านหลังของข้อสอบได้

-ทุจริตในการสอบมีโทษขั้นต่ำปรับตกในรายวิชานี้และพักการเรียน 1 ภาคการศึกษา-

ชื่อ-สกุล..รหัส...

1. จงแสดงวิธีการคูณเลขฐานสองแบบมีเครื่องหมายด้วยวิธี booth's Algorithm เมื่อกำหนด

```
Multiplicand =   0001 0111 (23)
Multiplier   =   1111 1001 (-7)
```

ให้แสดงรายละเอียดการคูณในแต่ละขั้นตอน (7 คะแนน)

A	Q	Q-1	M	อธิบาย

ชื่อ-สกุล...รหัส..

2. จงแสดงการหารเลขฐานสองแบบไม่มีเครื่องหมายเมื่อกำหนดให้

```
Dividend =  01 0110 0010  (354)
Divisor  =  00 0000 1111  (15)
```

พร้อมทั้งอธิบายการหารในแต่ละขั้นตอน (7 คะแนน)

M	A	Q	อธิบาย

3. จงอธิบายคุณสมบัติด้านต่างๆ เปรียบเทียบกันระหว่างสถาปัตยกรรมชุดคำสั่งแบบ CISC และ RISC (5 คะแนน)

คุณสมบัติ	CISC	RISC
ลักษณะของชุดคำสั่ง		
Addressing mode		
ความเร็วในการ fetch และ execute ชุดคำสั่ง		

4. กำหนดซีพียูเป็นแบบ 3-way superscalar และมีไปป์ลายน์ จำนวน 4 สเตทคือ fetch, decode, execute, writeback จงหาว่าหากจะรันโปรแกรมดังต่อไปนี้ให้มีประสิทธิภาพสูงสุดแล้ว จะต้องป้อนคำสั่งให้กับไปป์ลายน์ของซีพียูอย่างไร พร้อมทั้งวาดไดอะแกรมแสดงการไหลของคำสั่งผ่านไปป์ลายน์ของซีพียู (10 คะแนน)

```
I1:   ADD R1,R2  ;R1<= R1+R2
I2:   SUB R1,R3  ;R1<= R1-R3
I3:   AND R1,R4  ;R1<= R1 & R4
I4:   OR R3,R5
I5:   NOT R3
I6:   STORE [R3],R4
I7:   INC R4
I8:   ADD R4,R7
I9:   STORE [R4],R9
```

Clock1	Clock2	Clock3	Clock4	Clock5	Clock6	Clock7	Clock8	Clock9	Clock10

5. หากก่อนรันโปรแกรม ค่าในหน่วยความจำตำแหน่งที่ 13BE:0200-13BE:023F มีค่าดังต่อไปนี้

```
-d 200
13BE:0200  FF 02 03 04 05 06 07 08-09 0A 10 20 30 40 50 60
13BE:0210  70 80 90 A0 B0 C0 D0 E0-F0 00 00 00 00 00 00 00
13BE:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
13BE:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

จงแสดงค่าในรีจิสเตอร์และหน่วยความจำที่เปลี่ยนแปลงไปหลังจากรันโปรแกรมต่อไปนี้เสร็จสิ้นแล้ว

(10 คะแนน)

```
;กำหนดให้ค่าในรีจิสเตอร์ CS=13BE; DS=13BE; SS=13BE
13BE:0100 mov bp,200
13BE:0103 mov si,A
13BE:0106 mov di,14
13BE:0109 sub si,2
13BE:010C sub di,2
13BE:010F mov ax,[bp+si]
13BE:0111 add [bp+di],ax
13BE:0113 cmp si,0
13BE:0116 jnz 109
;กำหนดให้ตัวเลขที่แสดงทั้งหมดเป็นเลขฐานสิบหก
```

คำแนะนำ ให้ดูตารางคำสั่งของ 80x86 ท้ายข้อสอบประกอบการพิจารณา

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

6. จงอธิบายความแตกต่างของระบบการติดต่อหน่วยความจำของระบบมัลติโพรเซสเซอร์ UMA,NUMA และแบบ Distributed memory มาพอสังเขป (5 คะแนน)

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

โดย

A = 0100 0110 0111 1001 1010 1001 0000 0000
B = 0100 0011 1110 0011 0001 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)

.....

.....

.....

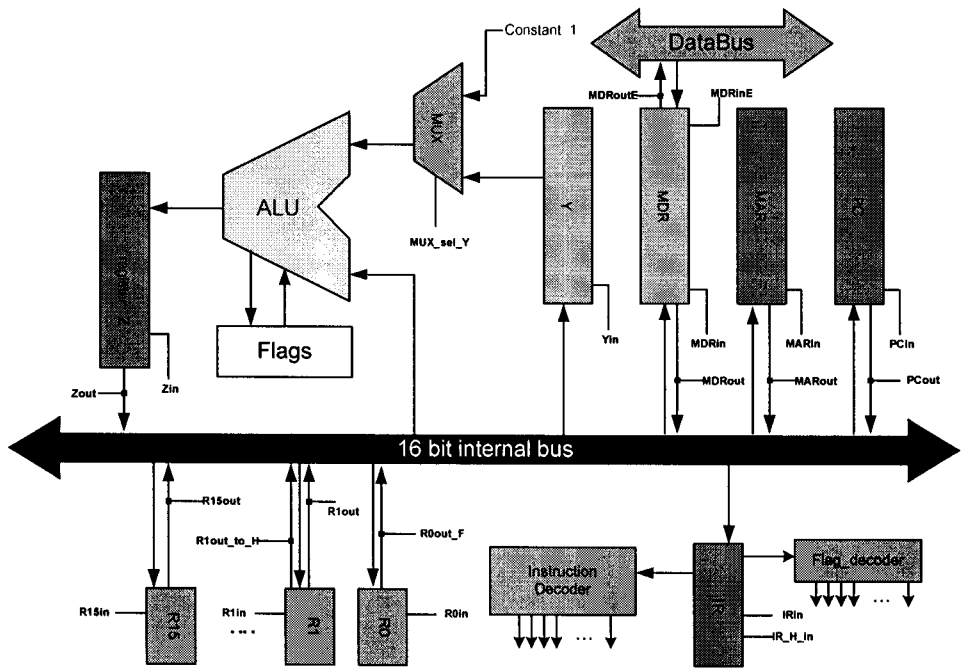
.....

.....

.....

.....

8. กำหนดให้ซีพียู Simple1 มีคำสั่งใช้งานจำนวน 16 คำสั่ง และมี datapath ดังรูป



หากซีพียูมีวงจร control unit แบบ hardwire แล้ว จงออกแบบวงจรสร้างสัญญาณควบคุมของซีพียูจำนวน 3 สัญญาณ คือ MARin, ALUsubb

<u>คำแนะนำ</u> ให้ดูภาคผนวก 1-3 ซึ่งอยู่ท้ายข้อสอบประกอบ

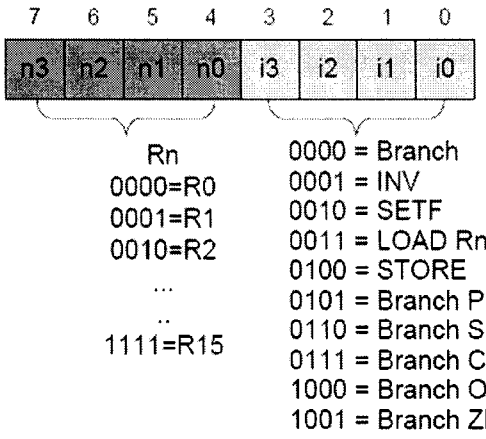
วงจรสร้างสัญญาณ ALUsubb (4 คะแนน)

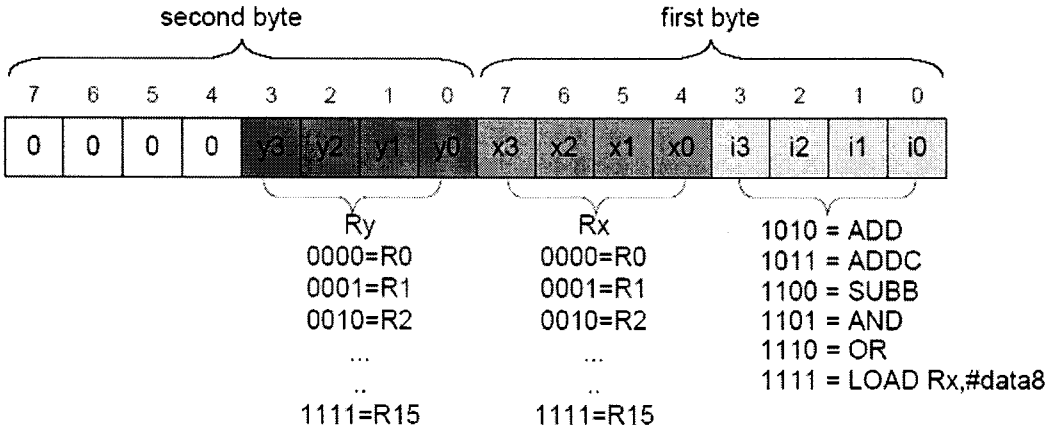
วงจรสร้างสัญญาณ MARin (6 คะแนน)

ภาคผนวก 1- ชุดคำสั่งของ Simple1

คำสั่ง	ความหมาย	ความยาวของคำสั่ง	ผลกระทบต่อแฟลก				
			CF	SF	ZF	OV	PF
INV Rn	กลับค่าบิตของรีจิสเตอร์ Rn เป็นตรงกันข้าม	1 byte			x		x
SETF b,v	เซ็ทค่าบิตของแฟลกที่ระบุให้มีค่าตามที่กำหนด เช่น SETF CF,1	1 byte	?	?	?	?	?
Load Rn,[R1R0]	ก๊อปปี้ค่าในหน่วยความจำตำแหน่งที่ระบุมาใส่ในรีจิสเตอร์ Rn	1 byte					
Store [R1R0], Rn	ก๊อปปี้ค่าในรีจิสเตอร์ Rn มาใส่ในหน่วยความจำตำแหน่งที่ระบุ	1 byte					
Branch [R1R0]	บรานช์แบบไม่มีเงื่อนไขไปยังแอดเดรสที่ระบุ	1 byte					
Branch ZF, [R1R0]	บรานช์ไปยังแอดเดรสที่ระบุหาก ZF=1	1 byte					
Branch SF, [R1R0]	บรานช์ไปยังแอดเดรสที่ระบุหาก SF=1	1 byte					
Branch PF, [R1R0]	บรานช์ไปยังแอดเดรสที่ระบุหาก PF=1	1 byte					
Branch OV, [R1R0]	บรานช์ไปยังแอดเดรสที่ระบุหาก OV=1	1 byte					
Branch CF, [R1R0]	บรานช์ไปยังแอดเดรสที่ระบุหาก CF=1	1 byte					
ADD Rx,Ry	Rx <= Rx+Ry (Addition)	2 bytes	x	x	x	x	x
ADDC Rx,Ry	Rx <= Rx+Ry+CF (Addition with carry flag)	2 bytes	x	x	x	x	x
SUBB Rx,Ry	Rx <= Rx – Ry – CF (Subtraction with borrow flag)	2 bytes	x	x	x	x	x
AND Rx,Ry	Rx <= Rx & Ry	2 bytes			x		x
OR Rx,Ry	Rx <= Rx | Ry	2 bytes			x		x
Load Rx,#data8	อ่านค่าคงที่ค่า #data8 มาใส่ในรีจิสเตอร์ Rx	2 bytes					

ภาคผนวก 2-ฟอร์แมตของคำสั่งของ Simple1





ภาคผนวก 3- ตัวอย่าง instruction cycle ของ Simple1

Instruction Fetch Cycle ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แอกตีฟในแต่ละคล็อกไซเคิล
0	MAR <= PC	PC_{out}, MAR_{in}
1	address_bus<= MAR, mem_read, Z<=PC+1, MDR<=mem[MAR]	PC_{out}, M_{read}, MDRinE, Z_{in}, ALU_{add},MUXselY=0
2	PC<=Z	PC_{in}, Z_{out}
3	IR<=MDR	IR_{in}, MDR_{out}

ไซเคิลการเอกซีคิวต์ของคำสั่ง ADDC Rx, Ry ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แอกตีฟในแต่ละคล็อกไซเคิล
4	MAR <= PC	PC_{out}, MAR_{in}
5	address_bus<= MAR, mem_read, Z<=PC+1, MDR<=mem[MAR]	PC_{out},M_{read}, MDRinE, Z_{in}, ALU_{add},MUXselY=0
6	PC<=Z	PC_{in}, Z_{out}
7	IR<=MDR	IR_H_{in}, MDR_{out}
8	Y<= Ry	Y_{in}, Ry_{out}
9	**Z<= Rx +Y+CF**	Rx_{out},MUXselY=1,ALU_{ADDC}, Z_{in}, CF_{in}, OV_{in}, ZF_{in},SF_{in},PF_{in}
10	**Rx<= Z**	Z_{out}, Rx_{in}, End

ไซเคิลการเอกซีคิวต์ของคำสั่ง Load Rx,#data8 ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แอกตีฟในแต่ละคล็อกไซเคิล
4	MAR <= PC	PC_{out}, MAR_{in}
5	address_bus<= MAR, mem_read, Z<=PC+1, MDR<=mem[MAR]	PC_{out},M_{read}, MDRinE, Z_{in}, ALU_{add},MUXselY=0
6	PC<=Z	PC_{in}, Z_{out}
7	Rx<= MDR	Rx_{in}, MDRout, End

ไซเคิลการเอกซีคิวต์ของคำสั่ง STORE [R1R0],Rn ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แอกตีฟในแต่ละคล็อกไซเคิล
4	MAR_H <= R1, MAR_L<= R0	MAR_{in}, R1_out_to_H, $R0_{out\_int\_bus}$
5	MDR <= Rn,	MDR_{in}, Rn_{out}
6	mem[MAR] <= MDR, mem_write	MDR_{outE}, Mwrite
7	deactivate write signal	MDR_{outE},End

ไซเคิลการเอกซีคิวต์ของคำสั่ง Branch [R1R0] ของ Simple1

คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แอกตีฟในแต่ละคล็อกไซเคิล
4	PC_H <= R1, PC_L<= R0	PC_{in}, R1_out_to_H, $R0_{out\_int\_bus}$,End

ไซเคิลการเอกซีคิวต์ของคำสั่ง Branch OV,[R1R0] ของ Simple1

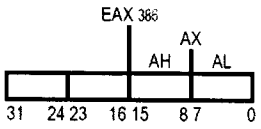
คล็อกไซเคิลที่..	Register Transfer Notation	สัญญาณควบคุมที่แอกตีฟในแต่ละคล็อกไซเคิล
4	if OV=0 then End	End_if_flag_not_set
5	PC_H <= R1, PC_L<= R0	PC_{in}, R1_out_to_H, $R0_{out\_int\_bus}$,End

ภาคผนวก 4 - 80x86 Instruction Set table

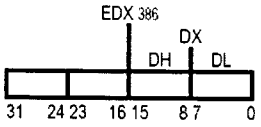
JUMPS (flags remain unchanged)							
Name	**Comment**	**Code**	**Operation**	**Name**	**Comment**	**Code**	**Operation**
CALL	Call subroutine	CALL Proc		RET	Return from subroutine	RET	
JMP	Jump	JMP Dest					
JE	Jump if Equal	JE Dest	(≡ JZ)	JNE	Jump if not Equal	JNE Dest	(≡ JNZ)
JZ	Jump if Zero	JZ Dest	(≡ 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	

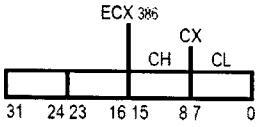
General Registers:



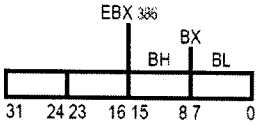
Accumulator



Data mul, div. IO



Count loop. shift



BaseX data ptr

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 0AFFFFh               ; define Doubleword, hex
S       DB "Hello !",0           ; 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: | - | - | - | - | O | D | I | T | S | Z | - | A | - | P | - | C |

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