รหัส นศ. _____ ชื่อ-สกุล _____ กอน ____

มหาวิทยาลัยสงขลานครินทร์
คณะวิศวกรรมศาสตร์

การสอบปลายภาค ภาคการศึกษาที่ ๒ ประจำปีการศึกษา ๒๕๕๐
วันที่ ๒๙ กุมภาพันธ์ พ.ศ. ๒๕๕๑ เวลา ๑๓๓๐-๑๖๓๐
วิชา 241-210 Microprocessor Architecture and The Assembly Language ห้อง R200

- ไม่อนุญาตให้นำเอกสารหรือเครื่องคำนวณเข้าห้องสอบ
- เขียนคำตอบให้อ่านง่าย ชัดเจน คำตอบข้อใดอ่านไม่ออก จะไม่ให้คะแนน
  ถ้าเนื้อที่สำหรับคำตอบไม่พอ อนุญาตให้เขียนต่อหน้าหลังของกระดาษคำตอบนั้น
- ข้อสอบมีทั้งหมด 7 หน้ารวมปก
- ข้อสอบมี 6 ข้อ คะแนนรวม 50 คะแนน ให้ทำทุกข้อ
- ให้เขียนชื่อ-สกุล รหัสนศ. **หน้าแรก และ ทุกหน้าที่เป็นกระดาษคำตอบ**

ข้อ	1	2	3	4	5	6	รวม
คะแนน							

ทุจริตในการสอบ โทษขั้นต่ำคือ ปรับตกในรายวิชานั้น
และพักการเรียน ๑ ภาคการศึกษา

รหัส นศ. _____ ชื่อ-สกุล _____ กอน ____

1. จงอธิบายความหมายของคำต่อไปนี้ (10 คะแนน)

1.1 Registers

1.2 Pipelines

1.3 RISC

1.4 CISC

1.5 Load-Store Architecture

2. จงเปรียบเทียบความซับซ้อนระหว่าง hardware และ software ของระบบคอมพิวเตอร์ที่ใช้สถาปัตยกรรมของโปรเซสเซอร์แบบ CISC และ RISC (3 คะแนน)

3. จงอธิบายความหมายของ conditional execution ภายใต้บริบทของสถาปัตยกรรมชุดคำสั่งของโปรเซสเซอร์ของ ARM (3 คะแนน)

รหัส นศ. _____ ชื่อ-สกุล _____ กอน ____

4. กำหนดให้ ค่าที่เก็บอยู่ใน registers ของ ARM และข้อมูลในหน่วยความจำ เป็นดังนี้

Reg	Value	Reg	Value	Memory Address	Value
R0	0x00000000	R8	0x80000000	0x80000000	0x01
R1	0x00000001	R9	0x90000000	0x80000001	0x02
R2	0x00000002	R10	0xA0000000	0x80000002	0x03
R3	0x00000003	R11	0x3FFFFFE0	0x80000003	0x04
R4	0x00000004	R12	0x3FFFFFF0	0x80000004	'A'
R5	0xFFFFFFFF	R13	0x3FFFFFF4	0x80000005	'B'
R6	0xFFFF0000	R14	0x3FFFFFF8	0x80000006	'C'
R7	0x0000FFFF	R15	0x40000000	0x80000007	'D'

จงแสดงผลลัพธ์ของการ execute คำสั่ง หรือ ชุดคำสั่งของ ARM ดังต่อไปนี้ ให้ผลของการ execute คำสั่งในแต่ละข้อ ไม่ขึ้นต่อกัน (ค่าข้อมูลตั้งต้นของ register และหน่วยความจำเหมือนเดิมตลอด)
(10 คะแนน)

4.1 sub r0, r1, r2

R0 = _____

4.2 add r0, r1, r2, LSL #4

R0 = _____

4.3 subs r0, r0, r1
adc r1, r1, #1

R0 = _____
R1 = _____

4.4 mov pc, lr

PC = _____
LR = _____

4.5 ldr r0, =0x80000000
ldr r1, [r0]

R0 = _____
R1 = _____

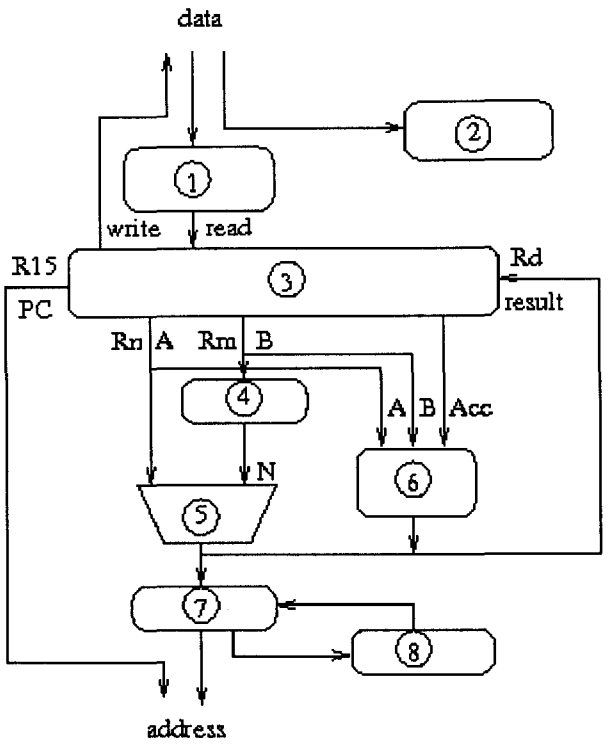
4.6 ldr r0, =0x80000000
ldrb r1, [r0, #4]!

R0 = _____
R1 = _____

_____ ____
_____ ____

รหัส นศ. _____ ชื่อ-สกุล _____ กอน ____

5. จาก dataflow model ของ ARM ในรูปต่อไปนี้ ให้ระบุชื่อ และฟังก์ชันการทำงานของ แต่ละ block (1) - (8) ดังแสดงในรูป dataflow diagram ของ ARM core (8 คะแนน)



1. _____

2. _____

3. _____

4. _____

5. _____

6. _____

7. _____

8. _____

รหัส นศ. _____ ชื่อ-สกุล _____ กอน ____

6. จงเติมส่วนที่ขาดหายไปของ โปรแกรมต่อไปนี้ (16 คะแนน)

6.1 delay subroutine (10 คะแนน)

```
@ delay - loop delay
@ input : R0 & R1
@     R0 = number of outer loop
@     R1 = number of inner loop
delay:
      _______     sp!, _______          @ save registers to stack
      mov         r2, r1                @ keep original value of r1 in r2
delay.0:
      _____       _____, _____          @ restore inner loop counter
delay.1:
      _____       r1, _____, _____      @ decrease r1
      bne         delay.1               @ inner loop
      subs        r0, _____, _____      @ decrease r0
      bne         delay.0               @ outer loop
      ldmea       sp!, {r0-r2}          @ restore registers from stack
      mov         pc, lr                @ return
```

6.2 getchar subroutine (6 คะแนน)

```
@ getchar - receive a character from serial port
@
@ return character in R0
getchar:
        stmfd   sp!, {r1-r3, _____}
        ______  r1, =U0LSR
        ______  r2, =U0THR
getchar.0:
        ldr     r3, [r1]                    @ wait for data in RXD
        _____   r3, #0x01
        beq     ______
        ldr     r0, [r2]                    @ read data to R0
        _____   sp!, {r1-r3, pc}
```

ฉัตรชัย จันทร์พริ้ม
กุมภ์ ๒๕๕๑

```
Syntax: MLA{<cond>}{S} Rd, Rm, Rs, Rn
        MUL{<cond>}{S} Rd, Rm, Rs
MLA             Rd = (Rm * Rs) + Rn
MUL             Rd = Rm * Rs

Syntax: <instruction>{<cond>}{S} RdLo, RdHi, Rm, Rs
SMLAL           [RdHi,RdLo] = [RdHi,RdLo] + (Rm*Rs)
SMULL           [RdHi,RdLo] = Rm*Rs
UMLAL           [RdHi,RdLo] = [RdHi,RdLo] + (Rm*Rs)
UMULL           [RdHi,RdLo] = Rm*Rs

Syntax: B{<cond>} label
        BL{<cond>} label
        BX{<cond>} Rm
        BLX{<cond>} label | Rm
B               pc = label
BL              pc = label, lr = next instruction
```

รหัส นศ. _____ ชื่อ-สกุล _____ กอน ____

ARM Instruction Set

```
Syntax: <instruction>{<cond>}{S} Rd, N
MOV             Rd = N
MVN             Rd = ~N

Syntax: <instruction>{<cond>}{S} Rn, N
CMN             flag set : Rn + N
CMP             flag set : Rn - N
TEQ             flag set : Rn ^ N
TST             flag set : Rn & N

Syntax: <instruction>{<cond>}{S} Rd, Rn, N
ADC             Rd = Rn + N + carry
ADD             Rd = Rn + N
RSB             Rd = N - Rn
RSC             Rd = N - Rn - !{carry}
SBC             Rd = Rn - N - !{carry}
SUB             Rd = Rn - N
AND             Rd = Rn & N
ORR             Rd = Rn | N
EOR             Rd = Rn ^ N
BIC             Rd = Rn & ~N

Syntax: MLA{<cond>}{S} Rd, Rm, Rs, Rn
       MUL{<cond>}{S} Rd, Rm, Rs
MLA             Rd = (Rm * Rs) + Rn
MUL             Rd = Rm * Rs

Syntax: <instruction>{<cond>}{S} RdLo, RdHi, Rm, Rs
SMLAL           [RdHi,RdLo] = [RdHi,RdLo] + (Rm*Rs)
SMULL           [RdHi,RdLo] = Rm*Rs
UMLAL           [RdHi,RdLo] = [RdHi,RdLo] + (Rm*Rs)
UMULL           [RdHi,RdLo] = Rm*Rs

Syntax: B{<cond>} label
       BL{<cond>} label
       BX{<cond>} Rm
       BLX{<cond>} label | Rm
B               pc = label
BL              pc = label, lr = next instruction
BX              pc = Rm & 0xFFFFFFFE, T = Rm & 1
BLX             pc = label, T = 1, lr = next instruction
                pc = Rm & 0xFFFFFFFE, T = Rm & 1, lr = next instruction

Syntax: <LDR|STR>{<cond>}{B}   Rd, addressing1
       LDR{<cond>}SB|H|SH     Rd, addressing2
       STR{<cond>}H           Rd, addressing2
LDR             Rd <- mem32[address]
STR             Rd -> mem32[address]
LDRB            Rd <- mem8[address]
STRB            Rd -> mem8[address]
LDRH            Rd <- mem16[address]
STRH            Rd <- mem16[address]
LDRSB           Rd <- SignExtend(mem8[address])
LDRSH           Rd <- SignExtend(mem16[address])
```

index method	Data	Base address	Example
Preindex+writeback	mem[base+offset]	base+offset	LDR r0,[r1,#4]!
Preindex	mem[base+offset]	not update	LDR r0,[r1,#4]
Postindex	mem[base]	base+offset	LDR r0,[r1],#4

```
Syntax: <LDM|STM>{<cond>}<addressing mode> Rn{!}, <register>{^}
LDM             {Rd}*N <- mem32[start adress + 4*N] optional Rn updated
STM             {Rd}*N -> mem32[start adress + 4*N] optional Rn updated
```

Addressing Mode	Description	Start Address	End Address	Rn!
IA	increment after	Rn	Rn+4*N-4	Rn+4*N
IB	increment before	Rn+4	Rn+4*N	Rn+4*N
DA	decrement after	Rn-4*N+4	Rn	Rn-4*N
DB	decrement before	Rn-4*N	Rn-4	Rn-4*N

รหัส นศ. _____ ชื่อ-สกุล _____ ตอน ____

Addressing Mode	Description	Pop	=LDM	Push	=STM
FA	full ascending	LDMFA	LDMDA	STMFA	STMIB
FD	full decending	LDMFD	LDMIA	STMFD	STMDB
EA	empty ascending	LDMEA	LDMDB	STMEA	STMIA
ED	empty descending	LDMED	LDMIB	STMED	STMDA

```
Syntax: SWP{B}{<cond>} Rd, Rm, [Rn]
SWP             tmp = mem32[Rn], mem32[Rn] = Rm, Rd = tmp
SWPB            tmp = mem8[Rn], mem8[Rn] = Rm, Rd = tmp

Syntax: SWI{<cond>} SWI_number
SWI             lr_svc = next instruction
                spsr_svc = cpsr
                pc = vectors + 8
                cpsr mode = svc
                cpsr I = 1

Syntax: MRS{<cond>} Rd, <cpsr|spsr>
        MSR{<cond>} <cpsr|spsr>_<fields>, Rm
        MSR{<cond>} <cpsr|spsr>_<fields>, #immediate
MRS             Rd = psr
MSR             psr{field} = Rm
MSR             psr{field} = immediate

Syntax: LDR Rd, =constant
        ADR Rd, label
LDR             Rd = 32-bit constant
ADR             Rd = 32-bit relative address
```

Condition Mnemonic

Mnemonic	Name
EQ	equal
NE	not equal
CS/HS	carry set/unsigned higher or same
CC/LO	carry clear/unsigned lower
MI	minus/negative
PL	plus/positive or zero
VS	overflow
VC	no overflow
HI	unsigned higher
LS	unsigned lower or same
GE	signed greater than or equal
LT	signed less than
GT	signed greater than
LE	signed less than or equal
AL	always(unconditional)