ภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
วิชา 01076115 ปฏิบัติการองค์ประกอบคอมพิวเตอร์
ภาคการศึกษาที่ 2/2564

รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

ลงชื่อตรวจ

_____

การทดลองที่ 6 Function Call

1. **Block Transfer**

ในบางครั้ง อาจมีความจำเป็นต้องจัดการหน่วยความจำ ขนาดหลายไบต์ หากจะเขียนโดยใช้ loop ก็จะทำให้ประสิทธิภาพไม่ดีเท่าที่ควร ผู้ผลิตโพสเซสเซอร์ ARM จึงได้สร้างคำสั่งสำหรับทำ block transfer โดยมีคำสั่งและรูปแบบดังนี้

LDM <option> (suffix) <Operand1>!, {<Registers>}
STM <option> (suffix) <Operand1>!, {<Registers>}

โดย Operand1 จะชี้ไปยังหน่วยความจำที่จะนำข้อมูลในรีจิสเตอร์ไปใส่
Registers คือ list ของรีจิสเตอร์ที่จะนำข้อมูลไปไว้ในหน่วยความจำ

เช่น

STM r0, {r1, r5-r8} หมายถึง นำ r1 -> [r0], r5 -> [r0+4], r6 -> [r0+8] r7 -> [r0+12]

ในคำสั่งข้างต้นนั้น ทุกครั้งที่นำเอาข้อมูลจากรีจิสเตอร์ไปใส่ในหน่วยความจำ จะมีการ + ตำแหน่งของหน่วยความจำขึ้น 4 แต่ยังสามารถใช้เงื่อนไขอื่นได้ด้วย โดยใส่ใน suffix

- IA = Increment After address += 4 หลังการ move
- IB = Increment Before address += 4 ก่อนการ move
- DA = Decrement After address -= 4 หลังการ move
- DB = Decrement Before address -= 4 ก่อนการ move

เช่น

STMDA r0, {r1, r5-r8} หมายถึง นำ r1 -> [r0], r5 -> [r0-4], r6 -> [r0-8] r7 -> [r0-12]

สำหรับเครื่องหมาย ! จะเป็นผลให้รีจิสเตอร์ชี้ตำแหน่งมีการ update ตำแหน่งหลังจากทำงาน
เช่น

STM r0, {r1, r3} หมายถึง นำ r1 -> [r0], r3 -> [r0+4] โดยเมื่อทำเสร็จ r0 จะมีค่าเดิม
STM r0!, {r1, r3} หมายถึง นำ r1 -> [r0], r3 -> [r0+4] โดยเมื่อทำเสร็จ r0 จะมีค่า +8

เราสามารถประยุกต์คำสั่งข้างต้นได้หลากหลาย เช่น สามารถนำไปบรรจุข้อมูลลงใน Stack ได้

เช่น จากเดิม

```
STR         lr, [sp, #-4]
STR         r0, [sp, #-4]
```

เป็น

```
STMDB       sp!, {lr, r0}
```

<u>หมายเหตุ</u> โปรแกรม assembler ได้ทำคำสั่งเทียม สำหรับ STMDB sp! และ LDMIA sp! โดยสามารถใช้คำสั่ง push และ pop แทนได้ เช่น

```
push        {lr, r0}
pop         {lr, r0}
```

นอกจากที่กล่าวมา ยังมี suffix อีก 4 ตัว ที่ออกแบบให้คำสั่ง block transfer ทำงานกับ stack ได้เหมาะสมมากขึ้น

- `STMFD/LDMFD  Full Descending Stack`
- `STMFA/LDMFA  Full Ascending Stack`
- `STMED/LDMED  Empty Descending Stack`
- `STMEA/LDMEA  Empty Ascending Stack`

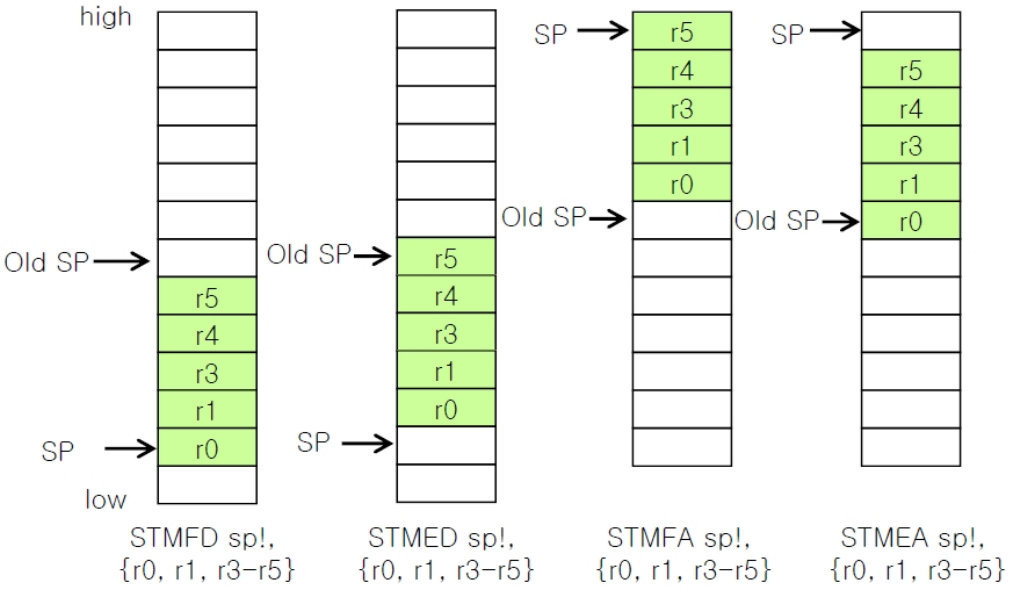
โดย

F คือ ทำงานจาก Address มากลงมา Address น้อย

E คือ ทำงานจาก Address น้อยขึ้นไป Address มาก

A คือ จะลดค่า sp

D คือ จะเพิ่มค่า sp



-

การทดลอง

1. ให้เขียนโปรแกรมภาษาซีต่อไปนี้

```
#include <stdio.h>

int main()
{
      putchar('*');
      return 0;
}
```

จากนั้นใช้คำสั่ง ดังนี้

gcc –S –o lab6a.s lab6a.c

2. ให้ดูไฟล์ lab6a.s ที่ถูกสร้างขึ้น ให้สรุปความแตกต่างระหว่าง กรณีที่เขียนโปรแกรม assembly ขึ้นเอง และกับที่เขียนด้วย c

_____
_____
_____
_____
_____
_____
_____
_____

โจทย์

ในหน่วยสืบราชการลับแห่งหนึ่ง ได้มีการกำหนดรหัสลับสำหรับใช้ในการสื่อสาร โดยตัวอักษรที่ใช้ในการสื่อสารแต่ละตัวจะถูกแทนด้วยตัวอักษรในวงล้อที่อยู่ถัดจากตัวอักษรนั้นไป 5 ตัวอักษร ในทิศทางทวนเข็มนาฬิกา จากวิธีการดังกล่าว ให้เขียนโปรแกรมเพื่อเข้ารหัสข้อความที่รับมา กำหนดให้ข้อความที่รับเข้ามาแต่ละครั้งมีความยาวไม่เกิน 80 ตัวอักษร โดยให้สร้างฟังก์ชันสำหรับ รับข้อมูล และแสดงผล โดยให้รับส่ง Parameter ให้ถูกต้อง (เก็บ register ทุกตัว ยกเว้น r0-r3)