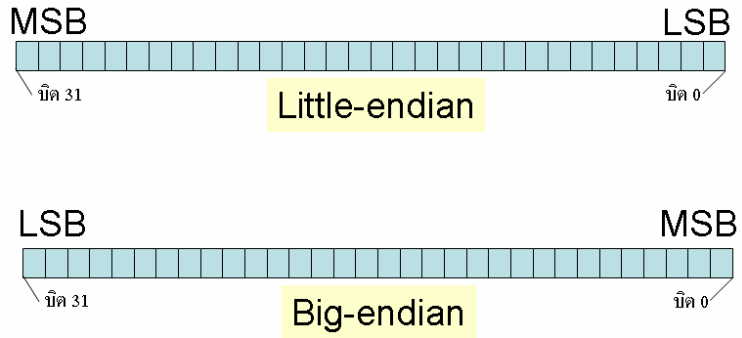
ชุดคำสั่งของอาร์มเซเว่น

โดย ศุภชัย บุศราทิจ
คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏเพชรบุรี
วันอังคารที่ ๓ ตุลาคม พ.ศ. ๒๕๔๕

สำหรับการเขียนโปรแกรมด้วยอร์มเซเว่นนั้นผู้เขียนมุ่งเน้นใช้ภาษาซี แต่เพื่อเป็นการทำความเข้าใจกับอาร์มเซเว่นนั้นจะมีประโยชน์ต่อการเขียนภาษาซีเป็นอย่างมาก ดังนั้น ในบทความนี้เราจะมาเรียนรู้เกี่ยวกับชุดคำสั่งภาษาแอสเซมบลีในระดับพื้นๆกันก่อน ซึ่งชุดคำสั่งของอาร์มเซเว่นนั้นจะมีด้วยกัน 2 ชุดคือ ชุดคำสั่งของอาร์มที่มีขนาด 32 บิตและชุดคำสั่งธัมบ์ (THUMB) มีมีขนาด 16 บิต

อาร์มเซเว่นเป็นหน่วยประมวลผลที่ถูกออกแบบขึ้นเพื่อประมวลผลได้ทั้งแบบบิ๊กเอนเดียน (big-endian) และลิตเติลเอนเดียน (little-endian) ที่มีความแตกต่างในเรื่องของการจัดเรียงบิตข้อมูล ดังรูปที่ 1 จะเห็นว่าในแบบบิ๊กเอนเดียนนั้นจะมองว่าบิต 0 คือ MSB และบิต 7 เป็น LSB แต่กรณีที่เป็นลิตเติลเอนเดียนจะมีการจัดเรียงแตกต่างกันไป คือ บิต 0 เป็น LSB และบิต 7 เป็น MSB แต่เพื่อความสะดวกผู้ออกแบบหน่วยประมวลผลมักเลือกเพียงอย่างใดอย่างหนึ่งเพื่อให้การทำงานของชิพมีประสิทธิภาพที่ดี ด้วยเหตุนี้ชิพตระกูล LPC2000 ของฟิลลิปจึงกำหนดเป็นการตายตัวเลยว่าใช้การเรียงบิตข้อมูลเป็นแบบลิตเติลเอนเดียนเท่านั้น



รูปที่ 1 การจัดเรียงบิตแบบลิตเติลและบิ๊กเอนเดียน

ชุดคำสั่งของอาร์มเซเว่นมีจุดหนึ่งที่น่าสนใจเป็นอย่างมาก คือ การออกแบบคำสั่งให้ทุกคำสั่งนั้นสามารถทำงานแบบเงื่อนไขได้ ซึ่งแตกต่างกับหน่วยประมวลผลแบบเดิมเป็นอย่างมาก เพราะในหน่วยประมวลผลแบบดั้งเดิมนั้นจะแยกชุดคำสั่งของการทำงานกับเงื่อนไขของการกระโดด (condition jump/branch) แต่ในอาร์มเซเว่นนั้นจะใช้ 4 บิตสุดท้ายของคำสั่งเป็นส่วนที่ใช้สำหรับเปรียบเทียบกับรหัสเงื่อนไขใน CPSR ถ้าไม่ตรงกันคำสั่งนั้นจะไม่ถูกทำงานและส่งการทำงานไปยังคำสั่ง NOP โดยเงื่อนไขที่ใช้นั้นจะถูกผนวกเข้ากับตัวนำหน้าชื่อคำสั่ง เช่น eqmov หมายถึงจะโอนข้อมูลเมื่อคำสั่งก่อนหน้านี้ทำให้ CPSR มีผลทำให้แฟล็ก Z มีค่าเป็น 1 เป็นต้น ซึ่งคำนำหน้าที่เป็นเงื่อนไขในการตรวจสอบก่อนทำงานมีรายละเอียดดังตารางที่ 1 ซึ่งการทำแบบ

นี้ทำให้การเขียนโปรแกรมมีลักษณะเลื่อนไหลไปในทางเดียวกัน เพราะไม่ต้องคอยตรวจสอบแฟล็กด้วยตนเอง แต่กำหนดไปเลยว่าถ้าก่อนหน้านี้เกิดผลอย่างที่กำหนดก็ให้ทำงานแต่ถ้าไม่ใช่ก็ไม่ต้องทำอะไร

กลุ่มคำสั่งของอาร์มเซเว่นถูกแบ่งเป็น 6 กลุ่มด้วยกันคือ การกระโดด การประมวลผลข้อมูล การถ่ายโอนข้อมูล การถ่ายโอนบล็อก การคูณ และการขัดจังหวะด้วยซอฟต์แวร์

ตาราง 1 เงื่อนไขที่เป็นคำนำหน้าคำสั่ง

คำนำหน้า	ผลกระทบกับแฟล็ก	ความหมาย
EQ	Z เป็น 1	เท่ากัน
NE	Z เป็น 0	ไม่เท่ากัน
CS	C เป็น 1	Unsigned higher or same
CC	C เป็น 0	Unsigned low
MI	N เป็น 1	เป็นค่าลบ
PL	N เป็น 0	เป็นค่าบวกหรือศูนย์
VS	V เป็น 1	เกิดการล้น (overflow)
VC	V เป็น 0	ไม่เกิดการล้น
HI	G เป็น 1 และ Z เป็น 0	Unsigned higher
LS	G เป็น 0 และ Z เป็น 1	Unsigned low or same
GE	N เท่ากับ V	มากกว่าหรือเท่ากัน
LT	N ไม่เท่ากับ V	น้อยกว่า
GT	Z เป็น 0 และ N เท่ากับ V	มากกว่า
LE	Z เป็น 1 และ N ไม่เท่ากับ B	น้อยกว่าหรือเท่ากัน
AL	ไม่สนใจใดๆ	เป็นจริงเสมอ

การกระโดด

คำสั่งกระโดดแบบพื้นฐานของอาร์มเซเว่นนั้นเป็นคำสั่งสำหรับกระโดดไปข้างหน้าหรือกระโดดกลับหลังได้สูงสุด 32 เมกะไบต์ ส่วนกรณีคำสั่งกระโดดที่มีการปรับปรุงที่เรียกกันว่าลิงก์กระโดด (BL: branch link) จะมีการทำงานเหมือนกับคำสั่งพื้นฐานแต่จะมีการจัดเก็บตำแหน่งของเรจิสเตอร์ PC บวกอีก 4 ไบต์ ลงในเรจิสเตอร์ลิงก์ (LR) ซึ่งมีประโยชน์ต่อการกลับมายังตำแหน่งของคำสั่งถัดที่จะต้องประมวลผล คำสั่งกระโดดมีดังนี้

B เป็นคำสั่งกระโดดไปยังตำแหน่งที่ต้องการ

BL เป็นคำสั่งกระโดดไปยังตำแหน่งที่ต้องการและเก็บค่าตำแหน่งเดิม + 4 เอาไว้ใน LR

BX เหมือนคำสั่ง B แต่เมื่อกระโดดไปแล้วจะกำหนดให้การทำงานเป็นโหมดธัมบ์

BLX เหมือนคำสั่ง BL แต่เมื่อกระโดดไปแล้วจะกำหนดให้การทำงานเป็นโหมดธัมบ์

การประมวลผลข้อมูล

รูปแบบของคำสั่งประมวลผลข้อมูลของอาร์มเซเว่นนั้นมีลักษณะดังนี้

เงื่อนไขของการทำงาน | รหัสการทำงาน | แฟล็กกำหนดเงื่อนไข | R1,R2,R3 | ค่าที่เติมให้ครบ 32 บิต

ตัวอย่างของคำสั่งประมวลผลข้อมูลได้แก่

AND กระทำกับบิตแบบแอนด์

EOR กระทำกับบิตแบบเอ็กซ์ออร์ (XOR: Exclusive OR)

SUB ลบ

RSB ลบส่วนกลับ (reverse subtract)

ADD บวก

ADC บวกโดยใช้แฟล็กแคร์รีประกอบการบวก

SBC ลบโดยใช้แฟล็กแคร์รีประกอบการลบ

RSC ลบส่วนกลับโดยใช้แฟล็กแคร์รีประกอบการทำงาน

TST ทดสอบ

TEQ ทดสอบความเท่ากัน

CMP เปรียบเทียบ

CMN Compare negated

ORR กระทำกับบิตแบบออร์

MOV ย้ายค่า

BIC ล้างค่าบิต

MVN Move negated

การถ่ายโอนข้อมูล

การถ่ายโอนข้อมูลเป็นคำสั่งถ่ายโอนข้อมูลระหว่างเรจิสเตอร์ที่เป็นที่เก็บข้อมูลกับแหล่งข้อมูลหรือแหล่งเก็บข้อมูลใหม่ ซึ่งคำสั่งในกลุ่มนี้ได้แก่

LDR เป็นการโหลดข้อมูลขนาด 32 บิตมาเก็บในเรจิสเตอร์

LDRH เป็นการโหลดข้อมูลขนาด 16 บิตมาเก็บในเรจิสเตอร์

LDRSH เป็นการโหลดข้อมูลขนาด 16 บิตแบบมีค่าลบมาเก็บในเรจิสเตอร์

LDRB เป็นการโหลดข้อมูลขนาด 8 บิตมาเก็บในเรจิสเตอร์

LDRSB เป็นการโหลดข้อมูลขนาด 8 บิตแบบมีค่าลบมาเก็บในเรจิสเตอร์

STR เป็นการนำข้อมูลขนาด 32 บิตจากเรจิสเตอร์ไปเก็บไว้ในแหล่งเก็บข้อมูล

STRH เป็นการนำข้อมูลขนาด 16 บิตจากเรจิสเตอร์ไปเก็บไว้ในแหล่งเก็บข้อมูล

STRSH เป็นการนำข้อมูลขนาด 16 บิตแบบมีค่าลบจากเรจิสเตอร์ไปเก็บไว้ในแหล่งเก็บข้อมูล

STRB เป็นการนำข้อมูลขนาด 8 บิตจากเรจิสเตอร์ไปเก็บไว้ในแหล่งเก็บข้อมูล

STRSB เป็นการนำข้อมูลขนาด 8 บิตแบบมีค่าลบจากเรจิสเตอร์ไปเก็บไว้ในแหล่งเก็บข้อมูล

นอกจากนี้ในอาร์มเซเว่นยังมีคำสั่ง swap สำหรับทำการสลับค่าระหว่างเรจิสเตอร์ 2 ตัว และคำสั่ง MSR สำหรับการอ่านข้อมูลจากเรจิสเตอร์ CPSR/SPSR มาเก็บในเรจิสเตอร์ทั่วไป กับ MRS ที่ทำหน้าที่นำข้อมูลจากเรจิสเตอร์ทั่วไปไปเก็บใน CPSR/SPSR แต่สองคำสั่งนี้จะไม่สามารถทำงานได้ในโหมดผู้ใช้

การถ่ายโอนบล็อก

เป็นการถ่ายโอนข้อมูลครั้งละหลายเรจิสเตอร์ โดยใช้คำสั่ง STM เพื่อนำค่าจะเรจิสเตอร์บล็อกต้นทางไปเก็บไว้ที่บล็อกปลายทาง และคำสั่ง LDM เพื่อนำค่าของเรจิสเตอร์บล็อกปลายทางมาเก็บในบล็อกต้นทาง

การคูณ

หน่วยประมวลผลของอาร์มเซเว่นนั้นได้ติดตั้งหน่วยสะสมการคูณหรือแม็ค (MAC: multiply accumulate unit) ซึ่งรองรับการคูณเลขจำนวนเต็มขนาด 32 และ 64 บิต โดยการคูณปกติคือการคูณตัวเลข 32 บิต 2 จำนวน แล้วนำผลลัพธ์ไปเก็บในเรจิสเตอร์ขนาด 32 บิตตัวที่ 3 แต่ถ้าเป็นการคูณแบบสะสม (multiply accumulate) จะทำเหมือนกันคูณปกติแต่จะนำผลลัพธ์บวกเข้ากับค่าที่เก็บในเรจิสเตอร์ปลายทาง สำหรับการคูณแล้วได้ผลลัพธ์เป็น 64 บิตนั้นจะอาศัยเรจิสเตอร์ปลายทาง 2 ตัวเป็นที่เก็บ

คำสั่งการคูณคือ

MUL เป็นคำสั่งคูณ ให้ผลลัพธ์เป็นจำนวนเต็มขนาด 32 บิต

MULA เป็นคำสั่งคูณแบบสะสม ให้ผลลัพธ์เป็นจำนวนเต็มขนาด 32 บิต

UMULL เป็นคำสั่งคูณของจำนวนเต็มแบบไม่มีค่าลบ ให้ผลลัพธ์เป็นจำนวนเต็มขนาด 64 บิต

UMLAL เป็นคำสั่งคูณสะสมของจำนวนเต็มแบบไม่มีค่าลบ และให้ผลลัพธ์เป็นจำนวนเต็มขนาด 64 บิต

SMULL เป็นคำสั่งคูณของจำนวนเต็มแบบมีเครื่องหมาย และให้ผลลัพธ์เป็นจำนวนเต็มขนาด 64 บิต

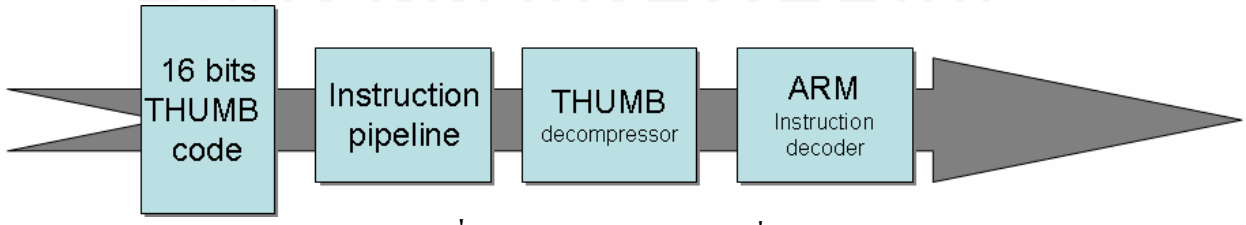
SMLAL เป็นคำสั่งคูณสะสมของจำนวนเต็มแบบมีเครื่องหมาย และให้ผลลัพธ์เป็นจำนวนเต็มขนาด 64 บิต

การขัดจังหวะด้วยซอฟต์แวร์

คำสั่ง SWI เป็นคำสั่งสำหรับเปลี่ยนให้ตัวประมวลผลทำงานเป็นโหมดซุปเปอร์ไวเซอร์ (supervisor mode) และกระโดดไปทำงานที่ตำแหน่ง 0x00000008 ซึ่งการใช้งานคำสั่งนี้สามารถกำหนดเงื่อนไขของการทำงานได้เหมือนกับคำสั่งอื่นๆ

ชุดคำสั่งธัมบ์

ในหน่วยประมวลแบบ 32 บิตของอาร์มเซเว่นนั้นจะมีชุดคำสั่งอีกชุดหนึ่งที่เป็น 16 บิตเรียกว่าธัมบ์ ซึ่งการทำงานของรหัสคำสั่งธัมบ์นั้นจะมีลำดับการส่งต่อของคำสั่งดังรูปที่ 2



รูปที่ 2 ลำดับการส่งต่อของคำสั่งธัมบ์

จากรูปที่ 2 จะเห็นได้ว่าชุดคำสั่งธัมบ์ขนาด 16 บิตนั้นจะถูกส่งเข่าท่อคำสั่ง (instruction pipeline) หลังจากนั้นจะถูกขยายให้เป็นขนาด 32 บิตด้วยธัมบ์ดีคอมเพรสเซอร์ (THUMB decompressor) หลังจากนั้นอาร์มเซเว่นจะทำการถอดรหัสชุดคำสั่งเพื่อดำเนินงานต่อไป

ข้อดีของการใช้ชุดคำสั่งเป็นแบบธัมบ์คือทำให้พื้นที่ของหน่วยความจำนั้นลดลงกว่า 30% แต่ว่าในการทำงานแล้วชุดคำสั่งของอาร์มแบบ 32 บิตนั้นจะทำงานได้เร็วกว่าถึง 40% เนื่องจากไม่ต้องเสียเวลาในการขยายชุดคำสั่งที่ถูกบีบอัดในแบบของ 16 บิตมาเป็น 32 บิตเพื่อทำการประมวลผล นั่นหมายความว่า ในความเป็นจริงแล้วชุดคำสั่งธัมบ์ก็คือชุดคำสั่งอาร์มที่ถูกบีบอัดเพื่อให้ขนาดของโปรแกรมนั้นเล็กลง เช่น คำสั่งของอาร์มเขียนว่า ADD R0,R0,R1 เมื่อเขียนเป็นธัมบ์จะถูกเขียนเป็น ADD R0,R1 แต่ให้ผลการทำงานเหมือนกันคือ R0 = R0+R1 จะเห็นว่าการเขียนนั้นสั้นกว่า

ในชุดคำสั่งของธัมบ์นั้นจะไม่สามารถเพิ่มส่วนของเงื่อนไขของการทำงาน และไม่สามารถใช้งานเรจิสเตอร์ได้ทั้งหมด นั่นคือในการใช้งานทั่วไปจะสามารถมองเห็นเฉพาะ R0 ถึง R7 แต่จะมีเพียงบางคำสั่งเช่น MOV, ADD และCMP ที่จะสามารถใช้งาน R8 ถึง R12 ได้ นอกจากนี้ธัมบ์จะไม่มีคำสั่ง MSR/MRS ให้ใช้งานอีกด้วย แต่ถ้าจำเป็นต้องใช้งานเราจะต้องสลับโหมดด้วยคำสั่ง BX และ BLX แต่อย่างไรก็ดี ทุกครั้งที่มีการเริ่มต้นทำงานใหม่หรือการรีเซ็ต (reset) นั้น อาร์มเซเว่นจะทำงานในโหมดอาร์มเสมอ

แต่สิ่งหนึ่งที่ธัมบ์มีให้ใช้งานแต่ไม่มีในอาร์มก็คือคำสั่ง PUSH/POP ที่ใช้สำหรับการนำค่าเข้าและออกจากหน่วยความจำสแตก

สรุป

จากบทความนี้ผู้อ่านคงพอมองเห็นภาพได้ว่าอาร์มเซเว่นนั้นมีชุดคำสั่งอะไรบ้างในการทำงาน ซึ่งการที่เราทราบว่าหน่วยประมวลผลมีคำสั่งใดหรือทำงานอะไรนั่นหมายความว่าเราก็จะทราบว่าหน่วยประมวลผลนั้นทำอะไรไม่ได้ด้วยเช่นกัน เช่น เราทราบว่าอาร์มเซเว่นสามารถบวก/ลบ/คูณ/กระทำแอนด์/กระทำออร์/กระทำเอ็กซ์คลูซีฟออร์ได้ แต่ไม่มีคำสั่งหาร เราทราบว่าอาร์มเซเว่นสามารถกระทำกับข้อมูลแบบจำนวนเต็มแบบมีค่าลบและไม่มีค่าลบได้ แต่ไม่สามารถกระทำกับข้อมูลแบบทศนิยมได้ เป็นต้น และด้วยข้อด้อยเหล่านี้เมื่ออยู่ในการเขียนด้วยโปรแกรมภาษาระดับสูงล้วนเป็นสิ่งที่เราไม่ต้องกังวลใดๆ เนื่องจากตัวแปลภาษาได้สร้างไลบรารีให้เราใช้งาน ซึ่งสะดวกกว่าการเขียนขึ้นเองเป็นอย่างมาก และไม่แปลกเลยที่ผู้สร้างเครื่องมือต่างๆล้วนทำการเปรียบเทียบศักยภาพของไลบรารีของตนกับผู้สร้างเครื่องมือค่ายอื่นๆ เพราะเป็นส่วนที่ทำให้ผู้ใช้เห็นถึงความคุ้มค่าในการลงทุนนั่นเอง เช่น ถ้าเราเลือกใช้ GCC-ARM7 ย่อมมีประสิทธิภาพด้อยกว่า Keil-ARM7 แต่ความแตกต่างคือตัวหนึ่งแจกฟรีแต่อีกตัวหนึ่งต้องซื้อ ถ้าของฟรีทำงานได้เป็นที่พึงพอใจก็ไม่มีปัญหา แต่ถ้าต้องการให้มีประสิทธิภาพในการทำงานสูงก็ต้องเลือกที่จะต้องซื้อเครื่องมือราคาแพง เป็นต้น

สุดท้ายต้องขอขอบคุณครอบครัวและทีมงานอีทีทีอีกเช่นเคยครับ โดยเฉพาะคุณกอบกิจ เติมผาติ ที่ได้อำนวยความสะดวกและช่วยเหลือเกื้อกูลผมเสมอมา ครั้งหน้าเราจะมาทำความรู้จักกับระบบเชื่อมต่อของอาร์มเซเว่นกันว่ามีอะไรบ้าง และสั่งงานได้อย่างไร หลังจากนั้นเราจะเริ่มศึกษาตัวอย่างการเขียนโปรแกรมกันเสียที