เขียนโปรแกรมกับ ARM7
ตอนที่ 1 แกนของ ARM7

ศุภชัย บุศราทิจ
(i_raek@hotmail.com)
คณะเทคโนโลยีสารสนเทศ
มหาวิทยาลัยราชภัฏเพชรบุรี

ในตระกูลของไมโครคอนโทรลเลอร์ LPC2000 จะใช้หน่วยประมวลผลกลางหรือซีพียู (CPU: **c**entral **p**rocessing **u**nit) ชนิด ARM7 เป็นตัวขับเคลื่อนการทำงาน ซึ่งในบทความชุดนี้จะเป็นการบทความในแนวของนักเขียนโปรแกรมเป็นหลัก นั่นคือ เน้นศึกษาวิธีการสั่งการหน่วยประมวลผล ARM7 แทนการเจาะลึกรายละเอียดของหน่วยประมวลผล

การที่จะเขียนโปรแกรมได้ดีนั้น ผู้เขียนควรจะเรียนรู้ถึงวิธีการทำงานและคุณสมบัติเด่นของหน่วยประมวลผลนั้นๆ เพื่อที่จะได้ใช้งานได้อย่างเหมาะสม โดยจากบทความที่แล้วผู้เขียนได้เกริ่นถึงบอร์ดไมโครคอนโทรลเลอร์ตระกูล ARM7 ของบริษัทอีทีทีไปแล้วนั้น คงทำให้เห็นภาพว่าคุณสมบัติ/วงจรการเชื่อมต่อ และวงจรสำหรับทดลองการทำงานของ ARM7 เป็นอย่างไร ขั้นตอนต่อไปที่จะต้องศึกษาก็คือวิธีการเขียนโปรแกรม ซึ่งการออกแบบชุดคำสั่งของ ARM7 นั้นเป็นชนิด RISC (**r**educe **i**nstruction **s**et **c**omputer) ที่มีการออกแบบให้มีคำสั่งขนาดเล็ก ทำให้สามารถประมวลผลได้เร็ว โดยเฉพาะใน ARM7 นี้ คำสั่งที่ออกแบบนั้นสามารถทำงานได้เสร็จในเพียง 1 รอบการทำงาน (single cycle) เท่านั้น

ไปป์ไลน์

ไปป์ไลน์ (pipeline) หรือการทำงานแบบสายท่อของ ARM7 นั้นมีการออกแบบไปป์ไลน์ของชุดคำสั่งเอาไว้ 3 สเตจ (stage) คือ ระยะของการอ่านชุดคำสั่ง (fetch) ระยะการถอดรหัสของชุดคำสั่ง (decode) และระยะของการทำงานตามชุดคำสั่ง (execute) ซึ่งการออกแบบด้วยวิธีการนี้มีข้อดีคือทำให้สามารถเรียกคำสั่งได้หลายคำสั่งมาซ้อนกันได้ ดังรูปที่ 1 จะเห็นว่าในช่วงแรกของการทำงานรอบที่ 1 นั้นจะมีการอ่านชุดคำสั่งที่ 1 เข้าสู่ไปป์ไลน์ หลังจากนั้นเมื่อทำการถอดรหัสชุดคำสั่งที่ 1 ระยะการอ่านจะว่างจึงทำการอ่านชุดคำสั่งที่ 2 เมื่อทำการทำตามชุดคำสั่งที่ 1 อยู่นั้นชุดคำสั่งที่ 2 ก็จะถูกถอดรหัสและชุดคำสั่งที่ 3 จะถูกอ่านเข้ามา และเมื่อทำไปเรื่อยๆจะเห็นว่าจะสามารถทำงานได้ 4 คำสั่งโดยใช้การทำงานเพียง 2 รอบเท่านั้น

	ชุดคำสั่งที่ 1	ชุดคำสั่งที่ 2	ชุดคำสั่งที่ 3	ชุดคำสั่งที่ 4
การทำงานรอบที่ 1	Fetch	รอ	รอ	รอ
	Decode	Fetch	รอ	รอ
	Execute	Decode	Fetch	รอ
การทำงานรอบที่ 2	ทำงานเสร็จ	Execute	Decode	Fetch
	ทำงานเสร็จ	ทำงานเสร็จ	Execute	Decode
	ทำงานเสร็จ	ทำงานเสร็จ	ทำงานเสร็จ	Execute
	ทำงานเสร็จ	ทำงานเสร็จ	ทำงานเสร็จ	ทำงานเสร็จ

รูปที่ 1 ไปป์ไลน์ของ ARM7 ทั้ง 3 สเตจ

เรจิสเตอร์

เรจิสเตอร์ (register) เป็นหน่วยความจำที่อยู่ในหน่วยประมวลผล และตัว ARM7 มีสถาปัตยกรรมการทำงานแบบ โหลดและสโตร์ (load and store) ซึ่งหมายความว่า ข้อมูลที่จะใช้ในการประมวลผลนั้นจะต้องถูกนำเข้ามาเก็บในเรจิสเตอร์แล้วจึงจะทำการประมวลผลและเมื่อได้ทำการประมวลผลเสร็จ ผลลัพธ์ที่ได้นั้นจะถูกนำไปเก็บเอาไว้ในเรสิเตอร์ ซึ่งสรุปการทำงานได้ 3 ขั้นตอนดังนี้

ขั้นตอนที่ 1 ทำการโหลดข้อมูลมาเก็บในเรจิสเตอร์

ขั้นตอนที่ 2 ทำการประมวลผลจากข้อมูลในเรจิสเตอร์

ขั้นตอนที่ 3 นำผลลัพธ์ที่ได้เก็บในเรจิสเตอร์ปลายทาง

เช่น ถ้าต้องการหาผลบวกของ M1 กับ M2 แล้วนำผลลัพธ์มาเก็บใน M3 สามารถนำมาเขียนเป็น 3 ขั้นตอนได้ดังนี้

ขั้นตอนที่ 1 ทำการโหลด M1 ไปเก็บในเรจิสเตอร์ (สมมุติว่าเป็น R1)

ทำการโหลด M2 ไปเก็บในเรจิสเตอร์ (สมมุติว่าเป็น R2)

ขั้นตอนที่ 2 เรียกคำสั่งบวก นั่นคือ ADD R3,R1,R2 (สมมุติว่าเก็บผลลัพธ์ใน R3)

คำสั่งนี้มีความหมายว่า R3 = R1 + R2

ขั้นตอนที่ 3 โอนข้อมูลจากเรจิสเตอร์ (R3) มาเก็บใน M3

เรจิสเตอร์สำหรับผู้ใช้ของหน่วยประมวลผล ARM7 นั้นจะมีขนาด 32 บิต ซึ่งมีทั้งหมด 17 ตัว คือ R0 ถึง R15 และ CPSR (**c**urrent **p**rogram **s**tatus **r**egister) โดยจำแนกกลุ่มทำงานได้ดังนี้

- R0 ถึง R12 สามารถใช้งานได้ตามที่ผู้ใช้หรือเขียนโปรแกรมต้องการ

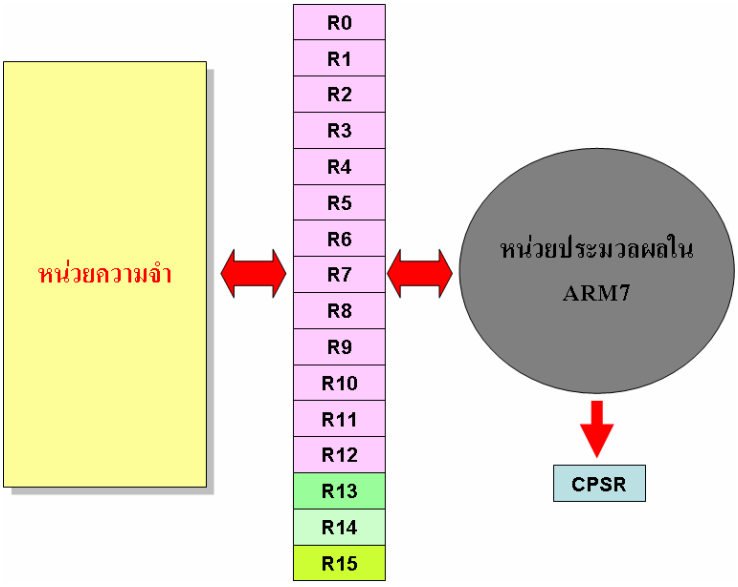
- R13 ถูกใช้เป็นที่เก็บค่าตัวชี้ไปยังหน่วยความจำสแตก (stack pointer)

- R14 หรือเรียกว่า LR (**l**ink **r**egister) ถูกใช้เป็นที่เก็บตำแหน่งของชุดคำสั่งถัดไปที่จะต้องประมวลผลก่อนที่จะเกิดการเรียกโปรแกรมย่อย (call) ซึ่งเป็นชุดคำสั่งที่จะต้องนำมาทำงานหลังจากที่โปรแกรมย่อยนั้นทำงานเสร็จแล้ว

- R15 หรือเรียกว่า PC (**p**rogram **c**ounter) ถูกใช้เป็นที่เก็บตำแหน่งของคำสั่งถัดไปที่จะถูกนำมาประมวลผล

- CPSR ใช้เป็นที่เก็บสถานการณ์ทำงานของคำสั่งที่ถูกประมวลผลไปหลังสุด ซึ่งนิยมเรียกเรจิสเตอร์นี้กันว่าเรจิสเตอร์สถานะหรือเรจิสเตอร์แฟล็ก (flag register)

จากเรจิสเตอร์ที่กล่าวมานี้ สามารถนำมาเขียนความสัมพันธ์ระหว่างกันได้ดังรูปที่ 2 ซึ่งจะเห็นว่าหน่วยความจำนั้นจะต้องถ่ายโอน/แลกเปลี่ยนข้อมูลกับเรจิสเตอร์ โดยการประมวลผลนั้นตะวประมวลผลจะใช้ข้อมูลจากเรจิสเตอร์ประกอบการทำงาน และเมื่อทำงานเสร็จจะบันทึกสถานะของการทำงานจากคำสั่งปัจจุบันเก็บเอาไว้ใน CPSR



รูปที่ 2 ความสัมพันธ์ระหว่างหน่วยความจำ เรจิสเตอร์และหน่วยประมวลผลของ ARM7

เรจิสเตอร์เก็บสถานะการทำงาน

เรจิสเตอร์เก็บสถานการณ์ทำงานของ ARM7 จะมีขนาด 32 บิต มีหน้าที่รายงานและใช้ควบคุมการทำงานของหน่วยประมวลผล ARM7 นั่นคือ เมื่อหน่วยประมวลผลทำงานเสร็จจะเก็บสถานการณ์ทำงานใน CPSR แล้วผู้เขียนโปรแกรมสามารถนำค่าเหล่านี้มาใช้เป็นการควบคุม

โปรแกรมที่เขียนเพื่อสั่งงานหน่วยประมวลผล หรืออาจจะกำหนดสถานะบางอย่างเพื่อให้หน่วยประมวลผลทำงานภายใต้ภาวะ (mode) ที่ผู้เขียนโปรแกรมกำหนด

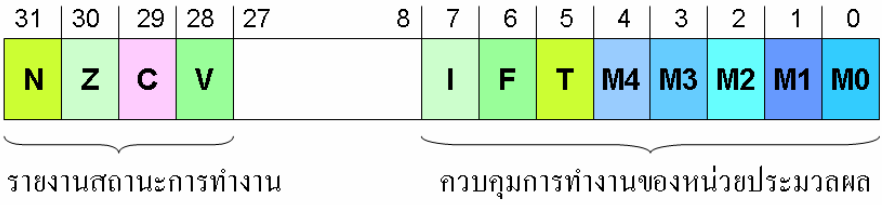
โครงสร้างของเรจิสเตอร์ตัวนี้เป็นดังรูปที่ 3 แบ่งได้ 2 กลุ่ม คือ

1. กลุ่มรายงานสถานการณ์ทำงาน ได้แก่

- N (**n**egative) เป็น 1 เมื่อผลการทำงานก่อให้เกิดค่าลบ
- Z (**z**ero) เป็น 1 เมื่อผลการทำงานทำให้เกิดค่าศูนย์
- C (**c**arry) เป็น 1 เมื่อผลการทำงานทำให้เกิดการทดค่า
- V (o**v**erflow) เป็น 1 เมื่อผลการทำงานเกิดการล้นของข้อมูล

2. กลุ่มกำหนดการทำงานของหน่วยประมวลผล ได้แก่

- I (**i**nterrupt enable) กำหนดให้เป็น 1 เมื่อต้องการดักสัญญาณขัดจังหวะแบบ IRQ
- F (**f**ast interrupt enable) กำหนดให้เป็น 1 เมื่อต้องการดักสัญญาณขัดจังหวะแบ FIQ
- T (**t**humb instruction set) กำหนดเป็น 1 เมื่อต้องการให้ทำงานชุดคำสั่งแบบ 16 บิต ซึ่งจะทำให้ ARM7 สามารถประมวลผลคำสั่งได้ 2 คำสั่งในการทำงานเพียง 1 รอบการทำงาน
- M4, M3, M2, M1, M0 ใช้ในการกำหนดภาวะการทำงาน ซึ่งมีทั้งหมด 7 โหมด (mode) ที่มีผลต่อการใช้งานเรจิสเตอร์แตกต่างกันไป แต่อย่างไรก็ดี ในการทำงานแบบโหมดผู้ใช้ (user mode) จะมีการทำงานเหมือนกับที่ได้อธิบายเอาไว้ในหัวข้อเรจิสเตอร์



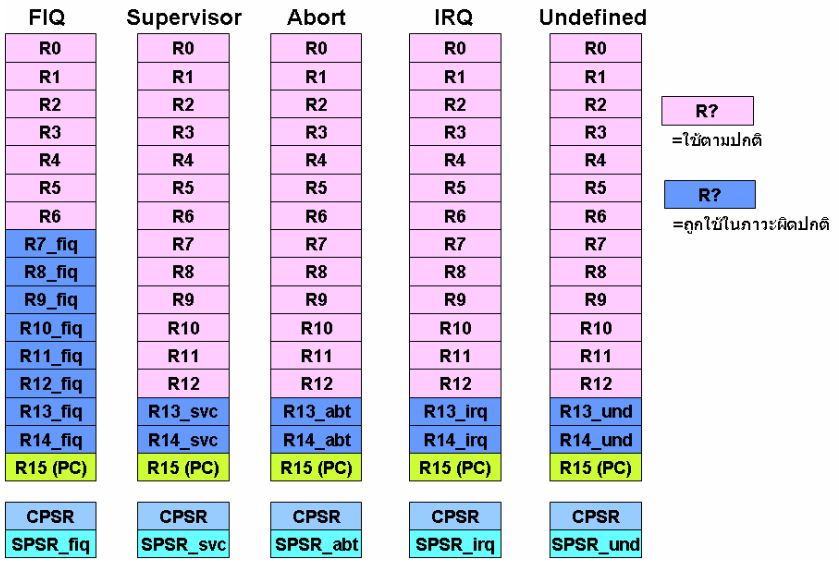
รูปที่ 3 ความหมายของบิตในเรจิสเตอร์สถานะการทำงาน

ภาวะผิดปกติ

ภาวะผิดปกติ (exception modes) เป็นการเกิดสิ่งที่ผิดปกติจากการทำงานทั่วไป และเมื่อเกิดสิ่งผิดปกตินี้หน่วยประมวลผลจะเปลี่ยนภาวะการทำงานของตัวเองพร้อมทั้งเปลี่ยนแปลงค่าของเรจิสเตอร์ PC เพื่อกระโดดไปทำงาน ณ ตำแหน่งที่กำหนดเอาไว้ดังตารางที่ 1 เพื่อตอบสนองต่อการ

ทำงานตามชนิดของภาวะผิดปกติที่เกิดขึ้น โดยแต่ละภาวะนั้นจะมีการใช้เรจิสเตอร์แตกต่างกันออกไปดังรูปที่ 4

สิ่งผิดปกติ	ภาวะการทำงาน	ตำแหน่ง
Reset	Supervisor	0x00000000
Undefined instruction	Undefined	0x00000004
Software interrupt (SWI)	Supervisor	0x00000008
Prefetch Abort (fetch memory abort)	Abort	0x0000000C
Data Abort (data access memory abort)	Abort	0x00000010
IRQ (interrupt)	IRQ	0x00000018
FIQ (fast insterrupt)	FIQ	0x0000001C



รูปที่ 4 การใช้เรจิสเตอร์ในภาวะผิดปกติ

เนื่องจากมีภาวะผิดปกติได้หลายประเภท ARM7 จึงมีการจัดลำดับความสำคัญเรียงจากมากไปน้อยเอาไว้ดังนี้

1. Reset
2. Data Abort
3. FIQ
4. IRQ
5. Prefetch Abort
6. Undefined instruction และ SWI

ผู้เขียนขออธิบายลำดับการทำงานเมื่อเกิดสิ่งผิดปกติเอาเป็นขั้นตอนดังนี้

1. อ่านตำแหน่งถัดไปที่จะเรียกขึ้นมาทำงานไปเก็บใน LR

2. คัดลอกสถานะใน CPSR ไปเก็บใน SPSR

3. นำค่าข้อมูลจากตำแหน่งที่ใช้ตอบสนองกับภาวะที่ผิดปกติไปเก็บใน PC ซึ่งหมายความว่าค่าที่เก็บในหน่วยความจำที่กำหนดตามตารางที่ 1 นั้นจะเป็นค่าตำแหน่งของชุดคำสั่งที่จะถูกเรียกใช้งานเมื่อเกิดสิ่งผิดปกติ

4. เริ่มทำงาน ณ ตำแหน่งที่ระบุใน PC

5. เมื่อทำงานชุดคำสั่งตอบสนองสิ่งผิดปกติเสร็จแล้วจะนำค่าที่เก็บใน LR มาเก็บใน PC เพื่อเตรียมทำงานชุดคำสั่งปกติ

6. นำค่าจาก SPSR มาเก็บใน CPSR เพื่อนำสถานการณ์ทำงานปกติกลับมา

7. เรียกคำสั่ง ณ ตำแหน่งที่เก็บใน PC มาทำงาน แล้วเลื่อน PC เป็นตำแหน่งของชุดคำสั่งถัดไป

สรุป

จากบทความนี้ผู้เขียนได้อธิบายการทำงานของหน่วยประมวลผล ARM7 เอาไว้ครอบคลุมเพียงพอต่อการทำความเข้าใจเพื่อใช้ประกอบการเขียนโปรแกรมในระดับเบื้องต้น ในบทความตอนต่อไปจะเป็นเรื่องของชุดคำสั่งของ ARM7 ซึ่งเป็นพื้นฐานสำคัญมากสำหรับคนที่ต้องการเขียนโปรแกรมภาษาซีบนระบบไมโครคอนโทรลเลอร์ ทั้งนี้ก็เนื่องจากการเขียนโปรแกรมบนระบบนี้อย่างไรก็ดีจะต้องมีความเข้าใจเรื่องการทำงานและชุดคำสั่งของหน่วยประมวลผลเพื่อใช้ในการเขียนโปรแกรม และแทบจะเป็นไปไม่ได้เลยที่จะเขียนโปรแกรมได้ดีโดยที่ไม่มีความรู้ใน 2 เรื่องนี้

สุดท้ายนี้ผู้เขียนขอขอบคุณครอบครัวที่ให้การสนับสนุนการทำงานของผู้เขียนมาตลอดถึงจะมีลุ่มๆดอนๆบ้างแต่ทุกอย่างก็คลี่คลายออกมาดีขึ้นเป็นลำดับ และที่ลืมไม่ได้คือ ทีมงานบริษัทอีทีที ที่ให้ความสนับสนุนบทความของผมมาตลอด โดยเฉพาะ คุณกอบกิจ เติมผาติ ที่คอยให้คำแนะนำและส่งเสริมการทำงานของผมเป็นอย่างดีและด้วยความอดทน (อย่างสูง) เพราะผู้เขียนเองมีงานประจำที่ค่อนข้างรัดตัว (ทั้งงานสอน สอบโครงการ เรียน วิจัย และครอบครัว) ทำให้การเขียนบทความและการปรับปรุงเว็บจึงเป็นไปอย่างคืบคลานทีละนิดทีละน้อยแต่ก็ได้พยายามให้เกิดความก้าวหน้าอย่างสม่ำเสมอเท่าที่เวลาจะอำนวย

บรรณานุกรม

Trever Martin. **The Insider's Guide to The Philips ARM7-Baesed Microcontrollers**. Hitex (UK), 2005.