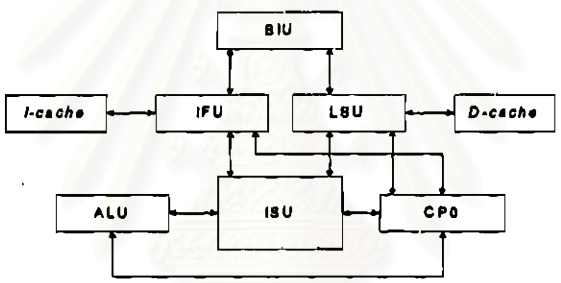
บทที่ 2
หลักการทำงานของ LSU

ในบทนี้จะกล่าวถึงหลักการทำงานของ LSU (Load / Store Unit) ซึ่งเป็นหน่วยประมวลผลที่สำคัญหน่วยหนึ่งในไมโครโพรเซสเซอร์ โดยจะกล่าวถึงตำแหน่งของ LSU ในไมโครโพรเซสเซอร์ และหน้าที่การทำงานของมัน รวมไปถึงหน่วยประมวลผลอื่นในไมโครโพรเซสเซอร์ที่ LSU ติดต่อด้วย

2.1 การออกแบบไมโครโพรเซสเซอร์

จากโครงงานวิจัยของนาย คาร์ลดิม สุวรรณมงคลและคณะ (พ.ศ. 2540) ได้ทำการออกแบบวงจรรวมเพื่อสร้างไมโครโพรเซสเซอร์ที่สามารถประมวลผลคำสั่งที่สำคัญของไมโครโพรเซสเซอร์ MIPS รุ่น R4000 ได้ โดยตัดส่วนของ Floating Point ออกไปเนื่องจากมีการทำงานที่ซับซ้อนมาก ในการออกแบบวงจรรวมดังกล่าว มีการแบ่งเป็นมอดูลย่อยๆ ดังรูปที่ 2.1



รูปที่ 2.1 Micorprocessor Block Diagram ของนาย คาร์ลดิม สุวรรณมงคลและคณะ

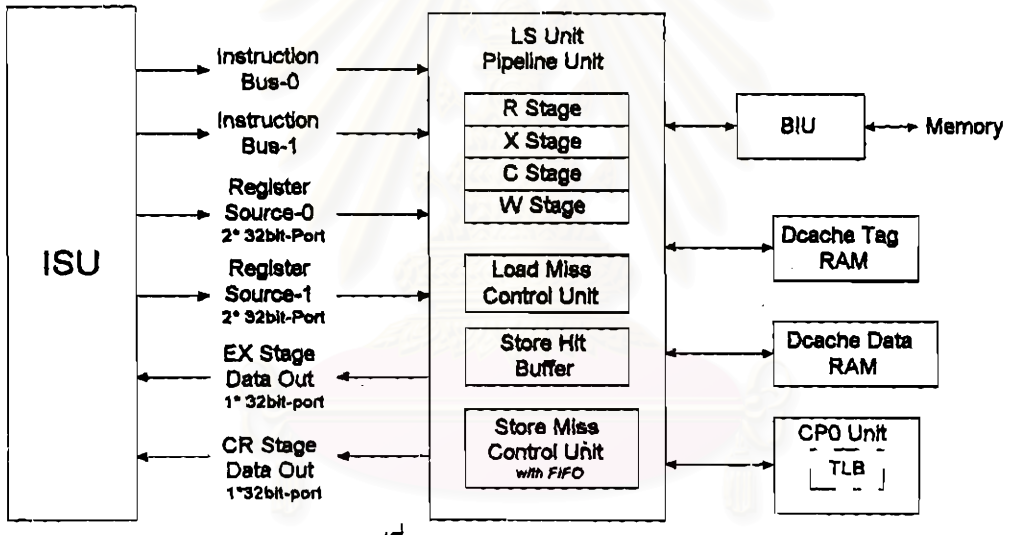
โดยมอดูลต่างๆ ในรูปที่ 2.1 มีหน้าที่ดังนี้

1. IFU (Instruction Fetch Unit) ทำหน้าที่ในการ Fetch คำสั่งมาจาก I-cache แล้วส่งไปให้กับ ISU เพื่อประมวลผลคำสั่งนั้นต่อไป
2. ISU (Instruction Schduler Unit) ทำหน้าที่ในการแจกจ่ายคำสั่งไปยังมอดูลอื่นๆ ตามคำสั่งที่ได้รับมา
3. CP0 (Coprocessor 0) ทำหน้าที่จัดการกับ exception และมีส่วนของ TLB (Table Lookaside Buffer) เพื่อใช้ในการเปลี่ยนค่า Virtual Address ให้เป็น Physical Address
4. LSU (Load / Store Unit) ทำงานตามคำสั่ง Load และ Store โดยติดต่อกับ D-Cache และ BIU ในการเข้าถึงหน่วยความจำ และส่งข้อมูลจากการ Load ไปให้กับ ISU
5. ALU (Arithmetic and Logic Unit) ทำงานในส่วนของการประมวลผลทางด้านคณิตศาสตร์ และส่งผลการคำนวณไปให้กับ ISU
6. BIU (Bus Interface Unit) ทำหน้าที่จัดลำดับความสำคัญของงานในการติดต่อกับหน่วยความจำหลัก
7. I-Cache (Instruction Cache) เป็นแคชสำหรับเก็บส่วนที่เป็นคำสั่ง
8. D-Cache (Data Cache) เป็นแคชสำหรับเก็บส่วนที่เป็นข้อมูล

งานวิจัยส่วนใหญ่มักมุ่งเน้นในเรื่องการประมวลผลการคำนวณทางด้านคณิตศาสตร์ (ALU) ว่าจะทำอย่างไรจึงทำให้การประมวลผลดังกล่าวเสียเนื้อที่ในส่วนของวงจรและเวลาที่ใช้ในการประมวลผลให้น้อยที่สุด แต่ประเด็นที่สำคัญอีกจุดหนึ่งที่ทำให้ความเร็วของเครื่องคอมพิวเตอร์ถูกจำกัด ก็คือ ในส่วนของการติดต่อกับหน่วยความจำ งานวิจัยนี้จึงมุ่งเน้นในการออกแบบส่วนของ LSU (Load / Store Unit) ซึ่งทำหน้าที่ประมวลผลคำสั่ง Load และคำสั่ง Store โดยมีการประมวลผลมากถึง 1 ใน 3 ของการประมวลผลทั้งหมด

2.2 หน้าที่ของ LSU

LSU ทำหน้าที่ในการประมวลผลคำสั่ง Load และ Store ซึ่งเป็นคำสั่งที่ใช้ในการติดต่อกับหน่วยความจำ โดยคำสั่งดังกล่าวและค่ารีจิสเตอร์ที่ใช้ในการประมวลผลจะถูกส่งมาจาก ISU (Instruction Scheduler Unit) และมี Data Cache เป็นหน่วยความจำระดับแรกที่ LSU ทำการร้องขอข้อมูล ข้อมูลที่ได้จากการทำงานในคำสั่ง Load จะถูกส่งกลับไปให้กับ ISU เพื่อทำการเขียนลงในรีจิสเตอร์ต่อไป โครงสร้างภายใน LSU ทำงานเป็น 4-stage pipeline และมีการติดต่อกับหน่วยประมวลผลอื่นดังรูปที่ 2.2



รูปที่ 2.2 LSU Block Diagram

LSU จะมีการส่งสัญญาณไปติดต่อกับหน่วยประมวลผลอื่น ได้แก่ Instruction Scheduler Unit (ISU), Bus Interface Unit (BIU), D-Cache Tag / Data และ Coprocessor0 (CP0) ซึ่งจะได้กล่าวถึงรายละเอียดของแต่ละหน่วยประมวลผลในหัวข้อต่อไป LSU ติดต่อสื่อสารกับ ISU ด้วยสัญญาณดังต่อไปนี้

1. Instruction Bus-0 และ Instruction Bus-1 เป็นคำสั่งขนาด 32 bit จำนวน 2 คำสั่งส่งเข้ามาพร้อมกับสัญญาณที่บอกให้ LSU ทราบว่า LSU จะต้องทำการประมวลผลในคำสั่งใด
2. Register Source-0 เป็นข้อมูลรีจิสเตอร์ขนาด 32 bit จำนวน 2 พอร์ต สำหรับทำงานตาม Instruction Bus-0
3. Register Source-1 เช่นเดียวกับ Register Source-0 แต่ทำงานตาม Instruction Bus-1
4. EX Strage Data Out เป็นข้อมูลขนาด 32 bit ที่ได้จากการทำงานตามคำสั่งบวก
   CR Stage Data Out เป็นข้อมูลขนาด 32 bit ที่ได้จากการทำงานตามคำสั่ง Load

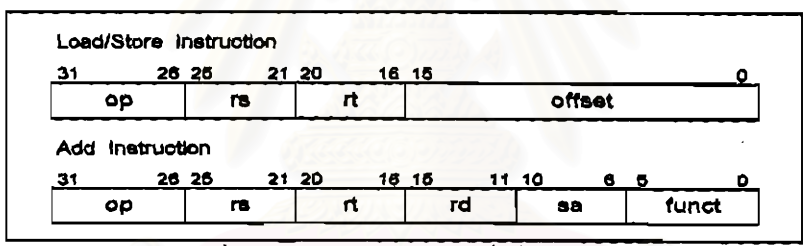
การประมวลผลภายใน LSU มีหน่วยประมวลผลย่อยที่สำคัญดังต่อไปนี้

1) Pipeline Stage Unit
2) Load Miss Control (LMC) Unit
3) Store Hit Control (SHC) Unit
4) Store Miss Control (SMC) Unit

2.2.1 Pipeline Stage Unit

Pipeline Stage Unit เป็นส่วนควบคุมการทำงาน Pipeline ภายใน LSU โดยแบ่ง Stage การทำงานออกเป็น 4 Stage ดังนี้

1) R Stage (Read Register File and Instruction Decoding) ทำการดึงอินพุตจาก Instruction Bus มาหลังจากได้รับสัญญาณจาก Instruction Scheduler Unit (ISU) โดย ISU เป็นผู้กำหนดว่าให้ดึงคำสั่งมาจาก Bus-0 หรือ Bus-1 การที่มี 2Bus นี้เพื่อให้สามารถรองรับการทำงานแบบ Superscalar ได้ แล้วจึงทำการ decode คำสั่งตามรูปที่ 2.3



รูปที่ 2.3 แสดงรูปแบบของคำสั่งที่ใช้ใน LSU

2) X Stage (eXecution) ภายในมี full 32-bit adder ไว้สำหรับหาตำแหน่งของ Virtual effective address สำหรับคำสั่ง Load และ Store โดยมีอินพุตหนึ่งมาจาก Register Source-0 และอีกอินพุตหนึ่งมาจาก Offset แล้วจะทำการส่งค่าดังกล่าวนี้ไปที่ CP0 เพื่อทำการแปลงเป็น Physical Address สำหรับในกรณีที่เป็นคำสั่งที่เกี่ยวกับการบวก เช่น ADD ก็สามารถคำนวณค่าได้ใน Stage นี้ โดย Stage นี้ได้รับอินพุตมาจาก Register Source ซึ่งมอดูล ISU ส่งมาให้ และผลการบวกที่ได้จะส่งออกไปที่ EX Stage Data Out เพื่อให้ ISU ทำการเขียนผลลัพธ์ที่ได้ลงในรีจีสเตอร์ต่อไป

3) C Stage (Cache read) ใน stage นี้ LSU จะรับค่า Tag จาก D-Cache Tag มาเปรียบเทียบกับค่า physical address ที่ได้จาก CP0 ว่าเท่ากันหรือไม่ ถ้าหากว่าค่า Tag กับค่าแอดเดรสที่ต้องการอ่านตรงกันและข้อมูลที่ได้รับนั้นมีอยู่จริง (valid) ก็จะให้ผลการอ่านแคชเป็น Hit และในทางตรงกันข้าม ถ้าหากค่า Tag กับค่าแอดเดรสไม่ตรงกันหรือข้อมูลที่อ่านได้ไม่มีอยู่จริง ก็จะให้ผลการอ่านแคชเป็น Miss ดังนั้นผลของการประมวลผลคำสั่ง Load จะแบ่งได้เป็น Load Hit และ Load Miss ส่วนผลของการประมวลผลคำสั่ง Store ก็จะแบ่งได้เป็น Store Hit และ Store Miss

- ในกรณีที่เกิด Load-Hit ขึ้น C Stage สามารถนำข้อมูลที่ได้มาประมวลผลตามคำสั่ง Load แล้วส่งค่าออกไปยัง CR Stage Data Out
- ในกรณีที่เกิด Load-Miss ขึ้น LSU จะส่งสัญญาณไปบอกที่ ISU ว่าข้อมูลที่ CR Stage Data Out นั้นไม่ถูกต้อง แล้วการประมวลผลของคำสั่ง load นั้น จะถูก Load Miss Control (LMC) Unit ควบคุมจนกว่าจะทำงานเสร็จ LSU จึงจะได้ข้อมูลที่ถูกต้องมาใช้งาน
- ในกรณีที่เกิด Store-Hit ขึ้น C Stage จะทำการเขียนข้อมูลที่ต้องการ Store ลงไปที่ Store Hit Buffer (SHB) และในกรณีที่โหมดการเขียนเป็นแบบ write through จะเขียนข้อมูลที่ต้องการ Store ลงใน FIFO ซึ่งอยู่ใน Store Miss Control (SMC) Unit ด้วย แต่ถ้าหาก SHB เต็ม จะทำให้เกิดการ "STALL" pipeline ซึ่งจะเป็นการบังคับให้ SHB ทำการเขียนข้อมูลที่เก็บเอาไว้ลงไปใน Data Cache จนกว่า SHB จะว่าง
- ในกรณีที่เกิด Store-Miss ขึ้น C Stage ส่งการประมวลผลไปที่ Store Miss Control (SMC) Unit ซึ่งจะมี FIFO ไว้สำหรับเก็บข้อมูลที่จะต้องการ Store ลงในหน่วยความจำหลัก แต่ถ้าหาก FIFO เต็ม จะทำให้เกิด "STALL" pipeline ซึ่งจะเป็นการบังคับให้ FIFO ทำการเขียนข้อมูลที่เก็บเอาไว้ลงไปในหน่วยความจำหลัก

จะเห็นว่าในกรณีของคำสั่ง Store จะให้เขียนข้อมูลพักเอาไว้ที่ Buffer และ/หรือ FIFO ก่อน เพื่อเป็นการไม่เสียเวลาในการติดต่อกับ Data Cache ในช่วงของ C Stage

4) W Stage (Write back) ใน stage นี้ ถ้าในหน่วย SHC มีข้อมูลที่ต้องการเขียนลง Data Cache เก็บอยู่ในบัฟเฟอร์ SHC จะเขียนข้อมูลลงใน Data Cache ได้ในกรณีที่ Data Cache ไม่มีการทำงานอื่นขอติดต่ออยู่ และในกรณีที่หน่วย SMC มีข้อมูลที่ต้องการเขียนลงหน่วยความจำอยู่ใน FIFO มันก็เขียนข้อมูลลงหน่วยความจำได้ ถ้าหากขณะนั้นไม่มีการทำงานใดขอติดต่อกับหน่วยความจำอยู่

2.2.2 Load Miss Control (LMC) Unit

หน่วย LMC มีการประมวลผลเมื่อเกิด Load Miss ขึ้น LMC จัดการติดต่อกับหน่วยความจำและนำข้อมูลที่ต้องการ Load เขียนลง D-Cache โดยการทำงานจะเริ่มพิจารณาจากบรรทัดของ D-Cache ที่จะถูกแทนที่ ว่าค่า WB bit* (Write Back bit) ถูกตั้งค่าเป็น '1' ไว้หรือไม่ ถ้า WB bit เป็น '1' แสดงว่าต้องนำข้อมูลจาก D-Cache เขียนลงในหน่วยความจำก่อนที่จะมีการแทนที่ข้อมูลใน D-Cache ซึ่งในงานวิจัยนี้ได้ออกแบบ WBB (Write Back Buffer) ไว้สำหรับเก็บข้อมูลที่จำเป็นต้องเขียนลงหน่วยความเพื่อลดการแย่ง Bandwidth ที่ต้องการติดต่อกับหน่วยความจำ แต่ถ้าค่า WB bit ='0' ก็สามารถเขียนข้อมูลทับลงไปได้เลย โดยในการเขียนข้อมูลลงใน D-Cache จะทำทั้งบรรทัด หรือทั้ง 4 doubleword

---

* สำหรับโหมดการเขียนแบบ Write Through จะไม่มีการตั้งค่า WB bit ดังนั้นเมื่อเกิด Load Miss ขึ้น จะไม่มีการเขียนข้อมูลลงใน WBB

2.2.3 Store Hit Control (SHC) Unit

หน่วย SHC จะถูกเรียกใช้ในกรณีที่เกิด Store Hit ขึ้น ภายในหน่วย SHC มีบัฟเฟอร์ (Store Hit Buffer – SHB) จำนวนหนึ่งใช้สำหรับเก็บข้อมูลที่ต้องการเขียนลงแคช บัฟเฟอร์ดังกล่าวช่วยลดการเกิดการ "STALL" pipeline อันเนื่องมาจากผลผลของการประมวลผล Store Hit ต้องการเขียนข้อมูลลงแคช และในขณะนั้นที่ Pipeline X Stage อาจมีความต้องการติดต่อกับแคช อันเนื่องมาจากการประมวลผลคำสั่ง Load หรือ Store อยู่ โดยข้อมูลภายใน SHB นี้จะถูกเขียนลง D-Cache เมื่อ D-Cache ว่าง (ไม่มีหน่วยใดร้องขอการเขียนหรืออ่าน D-Cache) และจะมีการตั้งค่า WB bit ภายใน D-Cache Tag ให้เป็น '1' ด้วย

สำหรับขนาดของ SHB ที่ผู้วิจัยได้ทำศึกษา โดยการปรับขนาดของ SHB เป็น 1, 2, 4 และ 8 บัฟเฟอร์ โดยแต่ละบัฟเฟอร์ทำการเก็บข้อมูลดังรูปที่ 2.4

Data[31:0]	Addr[12:2]	Bytep[3:0]	SetWB	Set1p
32	11	4	1	1

รูปที่ 2.4 รูปแบบการจัดเก็บข้อมูลของ SHB

Data[31:0] ใช้เก็บข้อมูลที่ต้องการเขียนลง D-Cache Data

Addr[12:2] ใช้เก็บค่าแอดเดรสเพื่อใช้เป็น index ในการชี้ D-Cache Data / Tag

Bytep[3:0] เป็นค่าที่แต่ละบิตระบุว่าต้องการเขียนข้อมูลลงไบต์ใด

SetWB เป็นค่าที่ใช้บอกความประสงค์ว่าต้องการตั้งค่า WB bit หรือไม่

Set1p เป็นค่าที่ใช่บอกความประสงค์ว่าต้องการเขียนข้อมูลลงแคชใน set ใด

2.2.4 Store Miss Control (SMC) Unit

หน่วย SMC จะถูกเรียกใช้ในกรณีที่เกิด Store Miss ขึ้น (ในโหมดการเขียนแบบ Write Through หน่วย SMC จะมีการประมวลผลต่อในกรณีที่เกิด Store Hit ด้วย) โดยภายในหน่วย SMC มี FIFO อยู่จำนวนหนึ่งใช้สำหรับเก็บข้อมูลที่ต้องการเขียนลงหน่วยความจำหลัก เพื่อลดการเกิด "STALL" pipeline อันเนื่องมาจากผลของคำสั่ง Store ที่ต้องการเขียนข้อมูลลงหน่วยความจำหลักในขณะที่หน่วย IFU หรือหน่วย LMC ต้องการอ่านข้อมูล หรือในขณะที่ WBB ต้องการเขียนข้อมูลกับหน่วยความจำหลักอยู่ โดยข้อมูลใน FIFO จะถูกเขียนลงหน่วยความจำหลักเมื่อหน่วยความจำหลักไม่มีการติดต่อจากหน่วยอื่น หรือในกรณีที่ FIFO ในหน่วย SMC เต็มแล้ว แต่ที่ Pipeline C Stage ต้องการเขียนข้อมูลลงใน FIFO อีก

สำหรับขนาดของ FIFO ที่ผู้วิจัยได้ทำศึกษา โดยการปรับขนาดของ FIFO เป็น 2, 4 และ 8 word และ doubleword โดยทำการเก็บข้อมูลดังรูปที่ 2.5 และ 2.6 ตามลำดับ

Addr[31:0]	Data[31:0]	Size
32	32	2

รูปที่ 2.5 รูปแบบการจัดเก็บข้อมูลของ FIFO แบบ word

Addr[31:0] เก็บค่าแอดเดรสเพื่อใช้เป็นตัวชี้ตำแหน่งในหน่วยความจำหลัก

Data[31:0] เก็บค่าข้อมูลที่ต้องการเขียนลงหน่วยความจำหลัก

Size[1:0] เก็บค่าจำนวนไบต์ที่ต้องการเขียนข้อมูลลงหน่วยความจำหลัก

Addr[31:0]	DataH[31:0]	DataL[31:0]	Size	DWp
32	32	32	2	1

รูปที่ 2.6 รูปแบบการจัดเก็บข้อมูลของ FIFO แบบ doubleword

Addr[31:0] เก็บค่าแอดเดรสเพื่อใช้เป็นตัวชี้ตำแหน่งในหน่วยความจำหลัก

DataH[31:0] เก็บค่าข้อมูลของ word หน้าที่ต้องการเขียนลงหน่วยความจำหลัก

DataL[31:0] เก็บค่าข้อมูลของ word หลังที่ต้องการเขียนลงหน่วยความจำหลัก

Size[1:0] เก็บค่าจำนวนไบต์ที่ต้องการเขียนข้อมูลลงหน่วยความจำหลัก

DWp เป็นค่าที่บอกใช้บอก BIU ว่าต้องการเขียนข้อมูลเป็น doubleword
(สำหรับ FIFO แบบ word ค่านี้จะเท่ากับ '0')

ข้อแตกต่างระหว่าง FIFO ชนิด word กับ doubleword ดูได้จากตัวอย่างคำสั่งต่อไปนี้ โดยสมมุติให้ผลของการประมวลผลคำสั่ง Store เป็น Miss และสมมุติให้ข้อมูลที่เก็บอยู่ในรีจีสเตอร์ R10 เป็น A0A0A0A0 และรีจีสเตอร์ R11 เป็น B0B0B0B0

```
Inst #1   SW      R10, B800(R0)
Inst #2   SW      R11, B804(R0)
```

ถ้าหากหน่วย SMC ใช้ FIFO ชนิด word การเก็บข้อมูลจะได้ผลดังรูปที่ 2.7 กล่าวคือ ผลจากการทำงาน SW แต่ละครั้งเก็บลง FIFO แต่ละบรรทัด

B800	A0A0A0A0	11
B804	B0B0B0B0	11

รูปที่ 2.7 ผลของการเก็บข้อมูลลง FIFO ชนิด word

ถ้าหากหน่วย SMC ใช้ FIFO ชนิด doubleword การเก็บข้อมูลจะได้ผลดังรูปที่ 2.8 กล่าวคือ ผลจากการทำงานในคำสั่ง Inst #2 มีการตรวจสอบค่าแอดเดรสดูว่าเป็นข้อมูลใน doubleword เดียวกับข้อมูลที่เก็บก่อนหน้านี้หรือไม่ ซึ่งค่าแอดเดรส B800 และ B804 เป็นข้อมูลใน doubleword เดียวกัน จึงเก็บข้อมูลลง FIFO ในบรรทัดเดียวกัน และตั้งค่า DWp = '1' ด้วย มีผลให้การเขียนข้อมูลลงในหน่วยความจำ ส่งไปเขียนเพียงครั้งเดียวแทนที่จะเป็น 2 ครั้งเหมือนอย่างการเก็บ FIFO แบบ word

B800	B0B0B0B0	A0A0A0A0	11	1

รูปที่ 2.8 ผลของการเก็บข้อมูลลง FIFO ชนิด doubleword

2.3 Instruction Schduler Unit (ISU)

ISU ทำหน้าที่เป็นตัวกลางในการประมวลผล โดยรับคำสั่งมาจากหน่วย IFU แล้วทำการถอดรหัสเพื่อตรวจสอบว่าควรส่งคำสั่งนั้นๆ ไปยังหน่วยประมวลผลใด นอกจากนี้ ISU ยังทำหน้าที่ในการติดต่อกับ General Purpose Registers (GPR) เพื่อขออ่านและเขียนค่ากับรีจีสเตอร์

นอกจากนี้ ISU ยังทำหน้าที่ควบคุมการทำงานของ Pipeline โดยการใช้สัญญาณ CANCEL, SLIP และ STALL ส่งไปบอกหน่วยประมวลผลอื่นๆ เพื่อให้แต่ละหน่วยประมวลผลควบคุมการทำงานของ Pipeline ให้ไปพร้อมๆ กัน โดยแต่ละสัญญาณมีความหมายดังนี้

1. CANCEL ทำการยกเลิกทุกคำสั่งที่ทำงานอยู่ในแต่ละ Stage ของ Pipeline และเริ่มต้นทำงานคำสั่งใหม่ตั้งแต่การ Fetch คำสั่ง
2. SLIP สัญญาณนี้เกิดขึ้นเมื่อเกิด Data Dependency กล่าวคือ มีการอ้างถึงข้อมูลในรีจีสเตอร์ตัวที่ยังประมวลผลไม่เสร็จ ดังตัวอย่างในรูปที่ 2.9 จะเห็นได้ว่าค่ารีจีสเตอร์ R10 ในคำสั่ง LW นั้นยังทำงานไม่เสร็จสิ้น คำสั่ง ADD ที่ตามมาต้องการนำค่ารีจีสเตอร์ R10 ไปใช้เป็นตัวบวก

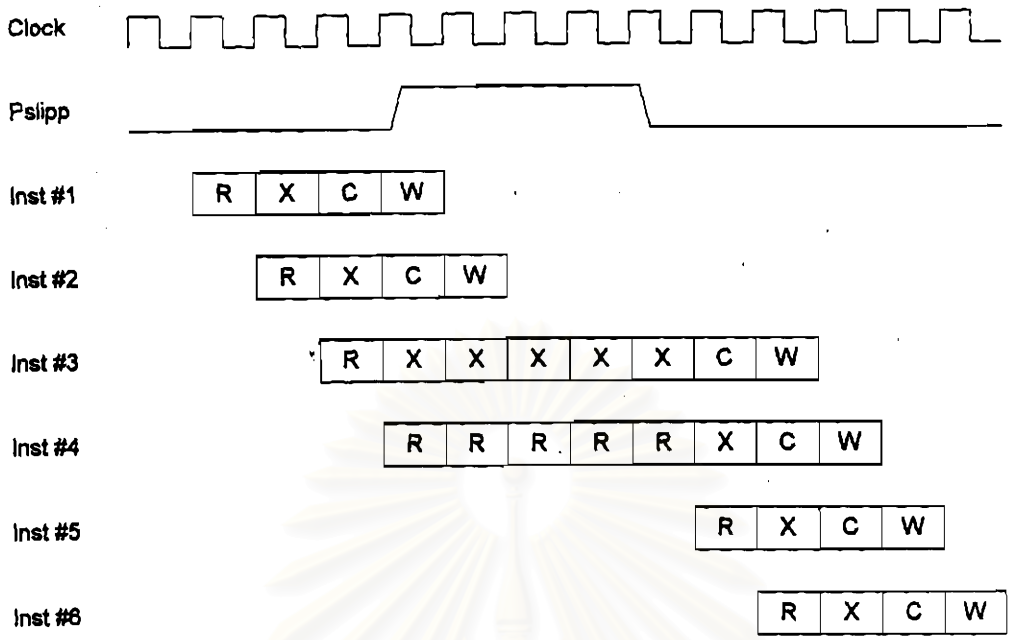
```
LW    (R10) B800(R0)
ADD   R12, R11,(R10)
```

รูปที่ 2.9 แสดงตัวอย่างการเกิด Data Hazard

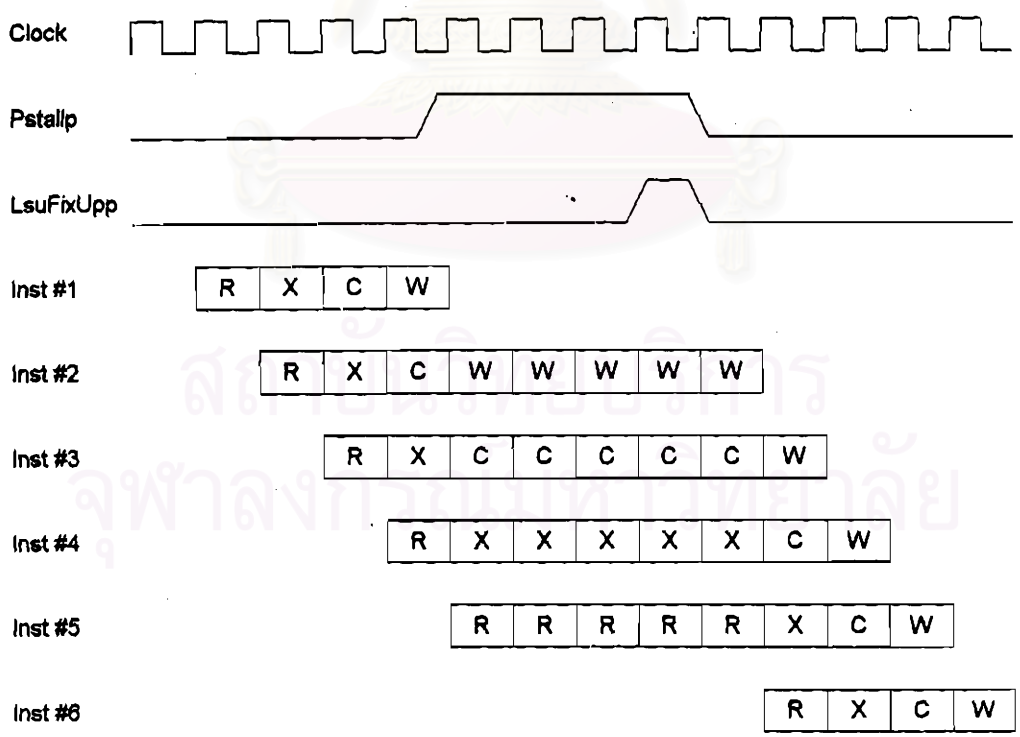
สัญญาณ SLIP มีผลทำให้การประมวลผลใน Pipeline R stage และ X stage ยังคงค้างอยู่เพื่อรอการอ่านข้อมูลที่ถูกต้อง แต่ที่ Pipeline C stage และ W stage ยังคงสามารถทำงานต่อไปได้ ดังรูป 2.10 จะเห็นได้ว่าเมื่อสัญญาณ Pslipp = '1' เกิดขึ้น การประมวลผลในคำสั่ง Inst #1 และคำสั่ง Inst #2 ยังคงประมวลผลต่อไปได้ แต่การประมวลผลในคำสั่ง Inst #3 และคำสั่ง Inst #4 ยังคงข้างการทำงานอยู่ที่ Pipeline X Stage และ R stage ตามลำดับ และเมื่อ Pslipp = '0' การทำงานของ Pipeline กลับสู่ภาวะปรกติ

3. STALL สัญญาณนี้เกิดขึ้นเมื่อเกิดภาวะที่ไม่สามารถอนุญาตให้ Pipeline ทำงานต่อไปได้ โดยภายในหน่วย LSU จะร้องขอการ "STALL" Pipeline เมื่อการทำงานในส่วน Load Miss, Store Hit หรือ Store Miss มีผลต่อการประมวลผลในคำสั่งที่กำลังทำงานอยู่ เช่น ขณะที่การทำงานในส่วน Load Miss ยังไม่เสร็จ ปรากฏว่าที่ Pipeline X Stage ต้องการประมวลผลคำสั่ง Load เราจึงจำเป็นต้องหยุดการทำงานของ Pipeline ก่อนเพื่อป้องกันการเกิด Load Miss ซ้อนกัน หรือในขณะที่บัฟเฟอร์ที่เก็บข้อมูลที่เกิดจากการทำงานของ Store Hit หรือ Store Miss เต็ม แล้วต้องการเขียนข้อมูลลงไปอีก เป็นต้น

สัญญาณ Pstallp มีผลต่อการทำงานของ Pipeline ดังรูป 2.11 กล่าวคือ เมื่อเกิด Pstallp = '1' ทุกการทำงานของ Stage Pipeline หยุดค้างอยู่กับที่จนกว่า Pstallp = '0' จึงสามารถทำงานได้ตามปรกติ



รูปที่ 2.10 การทำงานของ pipeline เมื่อเกิดเหตุการณ์ SLIP



รูปที่ 2.11 การทำงานของ pipeline เมื่อเกิดเหตุการณ์ STALL

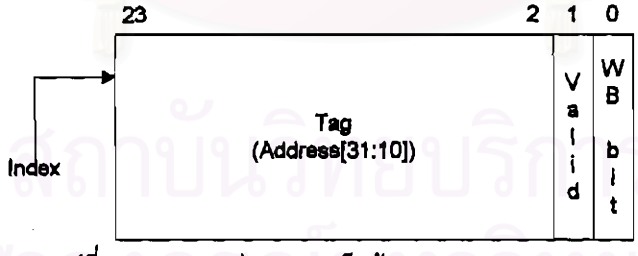
2.4 Bus Interface Unit (BIU)

ทำหน้าที่เป็นตัวจัดการในเรื่องลำดับความสำคัญสำหรับการติดต่อกับหน่วยความจำหลัก เพื่อให้เกิดการ "STALL" Pipeline น้อยที่สุด โดยมี LSU ทำการร้องขอการอ่านและเขียน และมี IFU ร้องขอการอ่าน การติดต่อหน่วยความจำถูกจัดลำดับความสำคัญเรียงตามหัวข้อต่อไปนี้

1. IFU ร้องขอการอ่านข้อมูลจากหน่วยความจำหลัก ซึ่งจะทำการขออ่านคราวละ 4 doubleword หรือคราวละ 8 คำสั่ง
2. LSU ร้องขอการอ่านข้อมูลจากหน่วยความจำหลัก จะเกิดขึ้นเมื่อเกิดการ load miss โดยจะทำการขออ่านข้อมูลคราวละ 4 doubleword
3. LSU ร้องขอการเขียนข้อมูลลงในหน่วยความจำหลัก การเขียนข้อมูลจะทำคราวละ 1 doubleword (หรือ byte, half word, word ขึ้นกับการขอเขียนในแต่ละกรณี) โดยจะสามารถเกิดขึ้นได้จาก 2 กรณี ได้แก่ จาก FIFO ในหน่วย SMC หรือจาก WBB (ซึ่งจะมีการกล่าวถึงในรายละเอียดต่อไป) โดยจะจัดลำดับความสำคัญย่อยเรียงไว้ดังนี้
   3.1 FIFO ในหน่วย SMC ขอเขียน
   3.2 WBB ขอเขียน

2.5 D-Cache Tag / Data RAM

D-Cache Tag ทำหน้าที่เก็บ Tag และ status bits ได้แก่ valid bit, write back bit เพื่อใช้ในการตรวจสอบการอ้างอิงตำแหน่งของหน่วยความจำ โดยรูปแบบการเก็บข้อมูลของ D-Cache Tag เป็นดังรูปที่ 2.12 และข้อมูลของ Tag ทั้ง 24 บิตมีความหมายดังนี้



รูปที่ 2.12 แสดงรูปแบบการเก็บข้อมูลของ D-Cache Tag

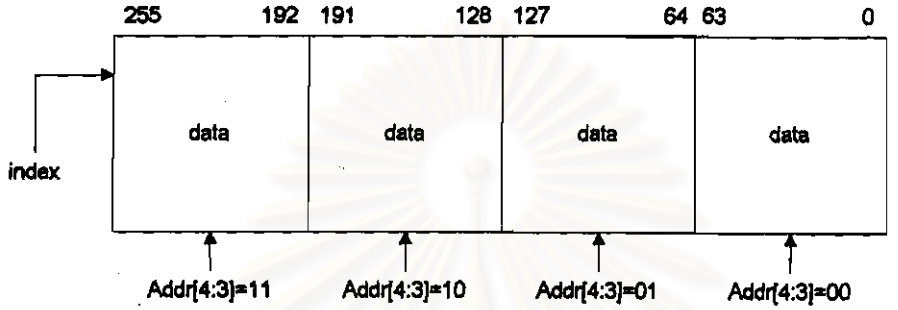
D-Cache Tag[23:2] ทำหน้าที่เก็บค่า Tag ซึ่งก็คือ Address[32:10] เพื่อใช้ในการตรวจสอบการอ้างอิงตำแหน่งของหน่วยความจำที่ต้องการติดต่อ โดยการนำค่า Tag ที่เก็บไว้นี้ไปเปรียบเทียบกับค่า Address [32:10] ของตำแหน่งข้อมูลที่ต้องการติดต่อกับหน่วยความจำ

D-Cache Tag[1] ทำหน้าที่เก็บค่า Valid bit เพื่อบอกว่าข้อมูลในบรรทัดเดียวกันนี้ของ D-Cache Data เป็นข้อมูลที่แท้จริงหรือไม่ โดยถ้าเป็นข้อมูลที่ใช้งานได้จริงจะกำหนดให้ Valid bit = '1'

D-Cache Tag[0] ทำหน้าที่เก็บค่า Write Back bit เพื่อบอกให้ทราบว่าข้อมูลใน D-Cache Data มีการเปลี่ยนแปลงไปจากเดิมหรือไม่ ถ้าหากมีการเปลี่ยนแปลง WB bit นี้จะถูกตั้งค่าเป็น '1' และเมื่อมีการถูก

แทนที่ ข้อมูลในบรรทัดนี้จำเป็นจะต้องเขียนข้อมูลกลับไปยังหน่วยความจำหลัก เนื่องจากข้อมูลในตำแหน่งนี้มีการเปลี่ยนแปลง สำหรับโหมดการเขียนที่เป็น Write Through นั้น ค่า WB bit ไม่ถูกใช้งาน เพราะทุกครั้งที่ทำงานตามคำสั่ง store จำเป็นต้องเขียนข้อมูลลงหน่วยความจำด้วย ดังนั้นค่าที่เก็บอยู่ใน D-Cache Data จะมีค่าตรงกับหน่วยความจำหลักอยู่แล้ว

D-Cache Data เป็นหน่วยความจำที่แทรกอยู่ระหว่างหน่วยประมวลผลกลางกับหน่วยความจำหลัก เพื่อเพิ่มความเร็วในการติดต่อกับหน่วยความจำ โดย D-Cache Data มีรูปแบบการเก็บข้อมูลดังรูปที่ 2.13



รูปที่ 2.13 แสดงรูปแบบการเก็บข้อมูลของ D-Cache Data

แต่ละบรรทัดของ D-Cache Data เก็บข้อมูลขนาด 4 doubleword การอ้างถึงข้อมูลในแต่ละ doubleword จะใช้ค่า Address[4:3] เป็นตัวชี้

Index ที่ใช้ในการชี้บรรทัดสำหรับ D-Cache Tag / Data คือ ค่า Address[n:5] โดยค่า n นี้จะขึ้นอยู่กับขนาดของแคชที่กำหนดในหน่วย CP0 ดังนั้น n จะเท่ากับ 9, 10, 11 และ 12 ตามขนาดของแคชที่เป็น 1KB, 2KB, 4KB และ 8KByte ต่อ 1 set

ค่า Status bit ใน D-Cache Tag ถูกนำไปใช้ดังนี้

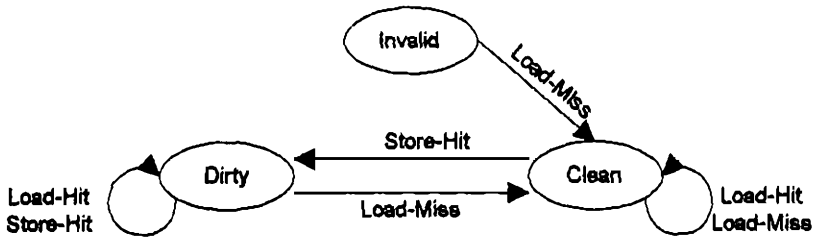
1) D-Cache State ใช้ในการบอกสถานะของแคชในแต่ละบรรทัด
2) Cache Lines Update ใช้ในการเลือก Set ที่จะถูกแทนที่

2.5.1 D-Cache state

งานวิจัยนี้สนับสนุนทั้งวิธีการเขียนแบบ Write Back และ Write Through โดยจะเก็บค่า State ไว้ในแต่ละบรรทัดของ D-Cache Tag สำหรับการเขียนแบบ Write Back จะมี 3 State ได้แก่ "Invalid", "Clean" และ "Dirty" แต่สำหรับการเขียนแบบ Write Through จะมีเพียงแค่ "Invalid" และ "Clean" เท่านั้น โดยตารางที่ 2.1 แสดงความหมายของแต่ละ Stage และการเปลี่ยน state ของโหมดการเขียนแบบ Write Back เป็นดังรูปที่ 2.14

State	V	WB	Condition
Invalid	0	X	Cache ในบรรทัดนั้นเก็บข้อมูลไม่จริงไว้
Clean	1	0	Cache ในบรรทัดนั้นเก็บข้อมูลจริง และตรงกับในหน่วยความจำ
Dirty	1	1	Cache ในบรรทัดนั้นเก็บข้อมูลจริง แต่ไม่ตรงกับในหน่วยความจำ

ตารางที่ 2.1 แสดง state ของแคชในแต่ละบรรทัด



รูปที่ 2.14 D-Cache write-back mode state diagram

เมื่อเริ่มทำงานในโหมดการเขียนแบบ Write Back ทุกบรรทัดใน D-Cache Tag จะถูกกำหนดค่าเป็น "Invalid" State และเมื่อเกิด Load Miss ขึ้นจะเปลี่ยนสถานะไปที่ "Clean" State ต่อมาถ้าหากเกิด Store-Hit ก็จะไปอยู่ที่ "Dirty" State และจะกลับไปอยู่ที่ "Clean" state เมื่อเกิด Load-Miss ขึ้น เพราะเป็นการ Refill ข้อมูลในตำแหน่งใหม่แล้ว สำหรับการเกิด Store Miss จะไม่มีผลต่อการเปลี่ยน State

การเปลี่ยน State ของโหมดการเขียนแบบ write-through เป็นดังรูปที่ 2.15 โดยในตอนเริ่มต้นของโหมดการเขียนแบบ Write Through ทุกบรรทัดใน D-Cache Tag จะเป็น "Invalid" State ทั้งหมด และเมื่อเกิด Load-Miss ขึ้นก็จะเปลี่ยนไปที่ "Clean" State และคงอยู่สถานะนั้นไม่เปลี่ยนแปลง

รูปที่ 2.15 D-Cache write-through mode state diagram

2.5.2 Cache Lines Update

ในงานวิจัยนี้ D-Cache สามารถจัดได้ทั้งแบบ direct-map และ 2-way set associative ด้วยการกำหนดค่าการจัดแบบ D-Cache ในหน่วย CP0 ซึ่งถ้าหากเป็นการจัด D-Cache แบบ 2-ways set associative ก็จำเป็นจะต้องมีกลวิธีที่ใช้ในการเลือกบรรทัดที่จะถูกแทนที่เมื่อเกิด Load-Miss ขึ้น โดย LSU จะทำการเลือกบรรทัดที่จะถูกแทนที่ด้วยวิธีการสุ่ม โดยค่าที่ได้จากการสุ่มจะเป็นตัวชี้เซตที่จะถูกแทนที่ โดยจะมีวิธีการเลือกเซตที่ถูกแทนที่ดังตารางที่ 2.2

งานวิจัยนี้ใช้วิธีการสุ่มทาง Software เนื่องจากในช่วงตรวจสอบความถูกต้องของการทำงานของวงจรจำเป็นต้องใช้ผลการสุ่มที่ได้เหมือนกันสำหรับแบบจำลองที่เขียนด้วยโปรแกรมภาษาซี และวงจรที่ออกแบบด้วยภาษา Verilog ซึ่งเราสามารถดัดแปลงเป็น Hardware ได้โดยง่าย อย่างเช่น การใช้ flip-flop ช่วย เป็นต้น

2.6 CP0 Unit

หน่วย CP0 ทำการติดต่อกับ TLB (Translation Lookaside Buffers) เพื่อเปลี่ยนค่า Virtual Effective Address ให้เป็นค่า Physical Address แต่ในงานวิจัยนี้ ได้ออกแบบให้ TLB เป็นเพียงกล่องดำ ดังนั้นค่า Virtual Effective Address ที่ TLB รับเข้ามา จะต่อตรงออกทางสัญญาณ Physical Address

ในงานวิจัยนี้ หน่วย CP0 ยังเก็บค่าพารามิเตอร์ต่างๆ ด้วย ได้แก่

1. cfgDS เป็นการกำหนดขนาดของ Data cache โดยจะสามารถมีค่าได้ตั้งแต่ 0 ถึง 3
2. cfgDE0 เป็นตัวกำหนดว่า Data cache ที่เซต 0 จะทำงานหรือไม่
3. cfgDE1 เป็นตัวกำหนดว่า Data cache ที่เซต 1 จะทำงานหรือไม่
4. CacheWBp เป็นตัวกำหนดวิธีการเขียนว่าให้เป็น Write Back (1) หรือ Write Through (0)

โดยการเซตค่า cfgDS, cfgDE0, cfgDE1 จะให้ผลดังตารางที่ 2.3

WB0	WB1	V0	V1	Random	Choose set
0	0	0	0	0	Set 0
0	0	0	0	1	Set 1
0	0	0	1	X	Set 0
0	0	1	0	X	Set 1
0	0	1	1	0	Set 0
0	0	1	1	1	Set 1
0	1	1	1	X	Set 0
1	0	1	1	X	Set1
1	1	1	1	0	Set 0
1	1	1	1	1	Set 1

ตารางที่ 2.2 แสดงการเลือกเซตในการถูกแทนที่

CfgDS	CfgDE0	cfgDE1	Data Cache
0	1	0	Direct Map 1KB
1	1	0	Direct Map 2KB
2	1	0	Direct Map 4KB
3	1	0	Direct Map 8KB
0	1	1	2-way Set 2KB
1	1	1	2-way Set 4KB
2	1	1	2-way Set 8KB
3	1	1	2-way Set 16KB

ตารางที่ 2.3 ขนาดของ Data Cache ที่กำหนดได้ในงานวิจัยนี้

2.7 สรุป

ในบทนี้ได้กล่าวถึงโครงสร้างของไมโครโพรเซสเซอร์ที่ออกแบบโดยนายวิเชียร สิริแสงทักษิณ และคณะ เพื่อแสดงถึงตำแหน่งของ LSU ในไมโครโพรเซสเซอร์ และได้กล่าวถึงการทำงานของหน่วย LSU ก่อนที่จะทำการออกแบบจริง รวมทั้งหน้าที่ของ ISU และสัญญาณที่ ISU ใช้ในการควบคุมการทำงานของ Pipeline อีกทั้งลำดับความสำคัญที่ IFU ใช้ในการติดต่อกับหน่วยความจำ นอกจากนี้ยังได้กล่าวถึงการเก็บข้อมูลภายใน D-Cache Data / Tag ความหมายของการเก็บ Status bit ทั้งโหมดการเขียนแบบ Write Back และ Write Through และวิธีการเลือกเซตที่จะถูกแทนที่ในกรณีที่การจัดแคชเป็นแบบ 2-way set associative และการเก็บค่ากำหนดขนาดของแคชในพารามิเตอร์ภายในหน่วย CP0