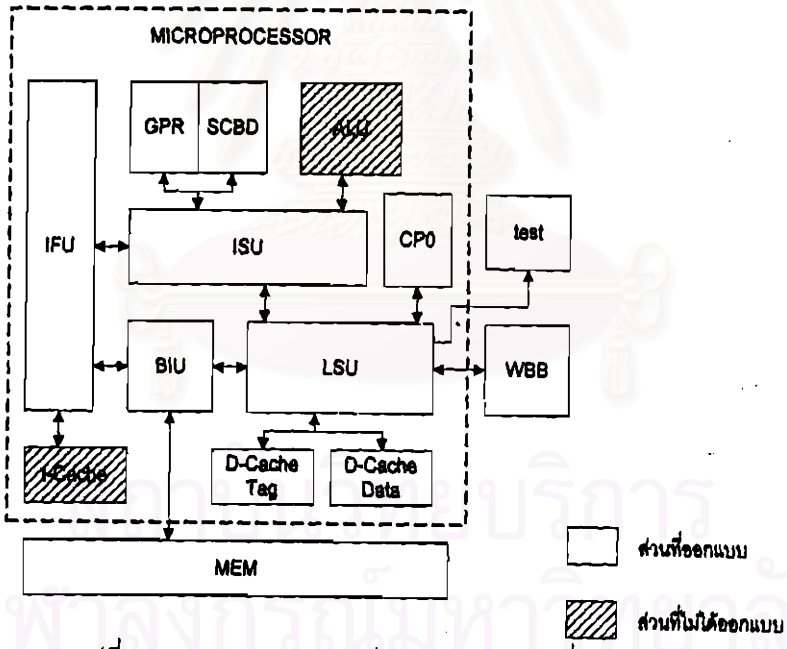
บทที่ 3
การออกแบบไมโครโพรเซสเซอร์

ในบทนี้จะกล่าวถึงการออกแบบไมโครโพรเซสเซอร์ที่ใช้ในงานวิจัยนี้ โดยจะแจกแจงในเรื่องของการทำงานและวิธีการออกแบบของแต่ละหน่วยประมวลผลที่อยู่ภายในไมโครโพรเซสเซอร์ และสัญญาณการเชื่อมต่อระหว่างหน่วยประมวลผล นอกจากนี้ยังได้กล่าวถึงขั้นตอนในการประมวลผลของชุดคำสั่งที่ใช้ในงานวิจัยนี้

3.1 หน่วยประมวลผลภายในไมโครโพรเซสเซอร์

การออกแบบไมโครโพรเซสเซอร์ที่ใช้ในงานวิจัยนี้มีลักษณะเป็น Block Diagram ดังรูปที่ 3.1 โดยได้ตัดส่วนประมวลผลที่เกี่ยวกับ Floating Point ออกไปเพื่อลดความซับซ้อนในการออกแบบวงจร และในงานวิจัยนี้เน้นเฉพาะการทำงานของคำสั่ง Load และ Store ในหน่วย LSU เท่านั้น ดังนั้นหน่วยประมวลผลที่ทำงานในคำสั่งอื่นจึงไม่ได้ถูกออกแบบในงานวิจัยนี้ ซึ่งได้แก่ ALU (Arithmetic Logic Unit) ที่ทำการประมวลผลในเรื่องของการคำนวณ หรือ I-Cache (Instruction Cache) ซึ่งจะได้กล่าวถึงเหตุผลในตอนต่อไป



รูปที่ 3.1 block diagram ของไมโครโพรเซสเซอร์ที่ออกแบบ

ในงานวิจัยนี้ได้ออกแบบหน่วยประมวลผลดังต่อไปนี้

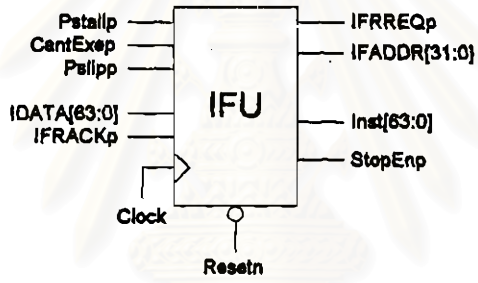
1. IFU	2. ISU	3. GPR	4. SCBD
5. BIU	6. CP0	7. MEM	8. WBB
9. D-Cache Tag	10. D-Cache Data	11. TEST	12. LSU

โดยได้เพิ่มในหน่วย TEST ขึ้นมาเพื่อทดสอบสมรรถนะของการประมวลผล ต่อไปนี้จะได้กล่าวว่าแต่ละหน่วยมีหน้าที่การทำงาน และการออกแบบอย่างไร

3.1.1 IFU (Instruction Fetch Unit)

IFU ทำหน้าที่จัดการในส่วนของการดึงคำสั่งมาจากหน่วยความจำมาประมวลผล โดยปรกติแล้วจะนำคำสั่งมาจาก Instruction Cache เนื่องจากพฤติกรรมในการประมวลผลของแต่ละโปรแกรมเป็นไปตามทฤษฎี Principal of locality ที่กล่าวถึงลักษณะการอ้างอิงข้อมูล 2 ลักษณะได้แก่ Temporal locality และ Spatial locality การอ้างอิงข้อมูลแบบ Temporal locality เป็นลักษณะการอ้างอิงถึงข้อมูลที่เพิ่งจะอ้างถึงไป เช่นการทำงานในส่วน loop ของโปรแกรมเป็นการดึงคำสั่งเดิมๆ ที่เคยได้ทำแล้วมาใช้ ส่วนการอ้างอิงข้อมูลแบบ Spatial locality กล่าวว่า ข้อมูลที่อยู่ใกล้เคียงกันจะมีโอกาสถูกอ้างถึงในเวลาอันใกล้ กล่าวคือมีการประมวลผลคำสั่งที่ทำงานต่อกันไป แต่วงจรที่ออกแบบในงานวิจัยนี้ ไม่สามารถประมวลผลคำสั่ง jump หรือ branch ได้ ดังนั้นจึงไม่มีความจำเป็นต้องใช้ I-Cache เพื่อเสริมคุณลักษณะ Temporal locality แต่ได้มีการออกแบบบัฟเฟอร์ขนาด 4 doublewords ตาม block size ของ data cache ที่เก็บบรรทัดละ 4 doublewords เพื่อเสริมคุณลักษณะ Spatial locality

สัญญาณที่ติดต่อกับ IFU เป็นดังรูปที่ 3.2



รูปที่ 3.2 สัญญาณที่ติดต่อกับ IFU

- สัญญาณเข้า

PstallP เป็นสัญญาณที่ ISU ส่งมาสั่งให้ pipeline หยุดและให้ส่งค่า Inst ที่ส่งไปให้ ISU เป็นค่าเดิม

CantExep เป็นสัญญาณที่ ISU ส่งมาให้ โดยเกิดขึ้นเนื่องจากการประมวลผลของ 2 คำสั่งที่ทำงานเป็น superscalar นั้น ใช้การประมวลผลในหน่วยเดียวกัน

PslipP เป็นสัญญาณที่ ISU ส่งมาให้ โดยเกิดขึ้นเนื่องมาจากการเกิด Data Hazard

IDATA[63:0] เป็นข้อมูลที่ได้จากการอ่านค่าจากหน่วยความจำ

IFRACKp เป็นสัญญาณที่ BIU ส่งมาให้ เพื่อบอกตอบรับการอ่านของ IFU จากหน่วยความจำ

- สัญญาณออก

IFRREQp เป็นสัญญาณร้องขอการอ่านข้อมูลจากหน่วยความจำ

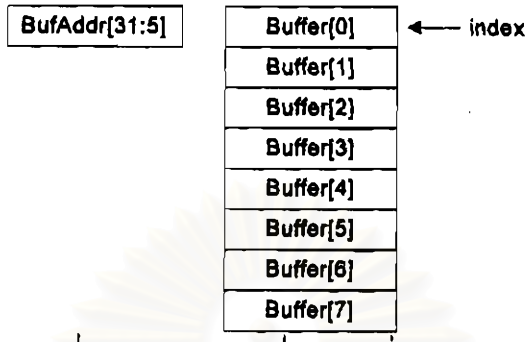
IFADDR[31:0] เป็นตำแหน่งที่ต้องการติดต่อกับหน่วยความจำ

Inst[63:0] เป็นคำสั่งที่ใช้ในการประมวลผล จำนวน 2 คำสั่งส่งไปให้ ISU

StopEnp เป็นสัญญาณสั่งหยุดการทดสอบ เกิดขึ้นเมื่อ IFU ได้รับคำสั่งสุดท้าย ซึ่งมีค่าเท่ากับ 0000000A (ผู้วิจัยระบุค่านี้เอง)

การทำงานของบัฟเฟอร์ภายใน IFU

บัฟเฟอร์ภายใน IFU มีขนาด 4 doublewords ตามขนาดของการอ่านข้อมูลจากหน่วยความจำ โดยบัฟเฟอร์ดังกล่าวสามารถเก็บคำสั่งได้ 8 คำสั่งดังรูปที่ 3.3



รูปที่ 3.3 แสดงบัฟเฟอร์ที่ใช้เก็บคำสั่งภายใน IFU

การตรวจสอบว่าคำสั่งในบัฟเฟอร์เป็นคำสั่งในตำแหน่งที่ต้องการหรือไม่ สามารถดูได้จากการตรวจสอบค่า PC(Program Counter) ตำแหน่ง [31:5] ว่าตรงกับ BufAddr ที่เก็บเอาไว้หรือไม่ ถ้าหากตรงกันก็สามารถนำคำสั่งจากบัฟเฟอร์ไปประมวลผลได้ แต่ถ้าหากไม่ตรงกัน จำเป็นต้องร้องขอมาจากหน่วยความจำโดยส่งสัญญาณ IFRREQp เป็น '1' และตำแหน่งของข้อมูลที่ต้องการติดต่อโดย IFRADDR มีค่าเท่ากับ PC

เมื่อได้รับข้อมูลมาทาง IDATA พร้อมกับสัญญาณ IFRACKp จากหน่วยความจำแล้ว IFU ทำการเขียนค่า PC ลงใน BufAddr และเก็บข้อมูลที่ได้รับจากหน่วยความจำลงในบัฟเฟอร์ทั้ง 8 ตัว

การกำหนดค่า PC

ในตอนเริ่มต้นการประมวลผล ค่า PC จะถูกกำหนดให้เท่ากับ 0000 0000 ต่อมาทุกขอบขาขึ้นของสัญญาณนาฬิกา จะมีการเปลี่ยนแปลงค่า PC โดยเรียงตามลำดับความสำคัญของการเกิดสัญญาณขึ้นดังนี้

เมื่อ	Pstallp = '1' or Pslipp = '1'	จะได้ค่า PC <= PC
	CantExep = '1'	จะได้ค่า PC <= PC + 4
	IOValidp = '1' and IEValidp = '1'	จะได้ค่า PC <= PC + 8
	IOValidp = '1' and IEValidp = '0'	จะได้ค่า PC <= PC + 4
	IOValidp = '0'	จะได้ค่า PC <= PC

โดย IOValidp เป็นค่าที่เกิดจากการตรวจสอบค่า PC[31:5] กับค่าใน BufAddr ถ้าหากว่าตรงกันก็จะได้ค่า IOValidp = '1' ถ้าไม่ก็จะมีค่าเป็น '0' ส่วน IEValidp เกิดจากการตรวจสอบค่า PC + 4 ตำแหน่ง [31:5] กับค่าใน BufAddr ว่าตรงกันหรือไม่

3.1.2 ISU (Instruction Scheduler Unit)

ภายในหน่วย ISU ทำงานเป็น 4-Stages Pipeline ได้แก่ R, X, C และ W stage โดย Stage ของ Pipeline จะเปลี่ยนเมื่อขอบขาขึ้นของสัญญาณนาฬิกา การทำงานของแต่ละ stage เป็นดังนี้

1) Read register and instruction decode stage

ใน Stage นี้หน่วย ISU ทำการกระจายคำสั่ง (InstO[31:0], InstE[31:0]) ไปให้กับหน่วยอื่นๆ พร้อมทั้งมีสัญญาณควบคุมบอกให้หน่วยต่างๆ ได้รู้ว่าหน่วยนั้นๆ ต้องประมวลผลคำสั่งใด แต่ในงานวิจัยนี้หน่วย ISU จะส่งสัญญาณไปให้เฉพาะหน่วย LSU เท่านั้น โดยสัญญาณ LsuOEnp สั่งให้หน่วย LSU ประมวลผลคำสั่ง InstO และสัญญาณ LsuEEnp สั่งให้ประมวลผลตามคำสั่ง InstE ถ้าหากว่าทั้งสัญญาณ LsuOEnp และ LsuEEnp มีค่าเป็น '1' พร้อมกัน หน่วย ISU จะส่งสัญญาณ CantExep ไปบอกหน่วย IFU ทราบ นอกจากนี้ หน่วย ISU ยังส่งสัญญาณไปขออ่านค่ารีจีสเตอร์ที่หน่วย GPR โดยตำแหน่งของรีจีสเตอร์ที่ส่งไปนั้น ได้จากการถอดรหัสคำสั่งตามภาคผนวก ก. และส่งสัญญาณไปให้หน่วย SCBD เพื่อตรวจสอบการอ่านค่ารีจีสเตอร์ และตรวจสอบด้วยว่าคำสั่งที่ได้ต้องมีการเขียนค่ากลับไปยังรีจีสเตอร์หรือไม่

2) eXecution stage

การทำงานใน Stage นี้ ISU จะทำการตรวจสอบค่าที่ได้จากการอ่านรีจีสเตอร์จากหน่วย GPR (GPRegS1, GPRegT1, GPRegS0 และ GPRegT0) แล้วส่งต่อไปยังหน่วยอื่นที่ต้องการ ซึ่งในงานวิจัยนี้ ISU จะส่งไปให้หน่วย LSU เท่านั้น ถ้าหากเกิดเหตุการณ์ stall pipeline ขึ้น ก็จำเป็นต้องปล่อยค่าเก่าไปให้กับหน่วย LSU พร้อมกันนี้ยังได้ตรวจสอบค่าจากตารางในหน่วย SCBD (ได้แก่ S1bit, T1bit, S0bit, T0bit) ด้วยว่ารีจีสเตอร์ตัวที่ต้องการอ่านนั้นอ่านได้หรือไม่ ถ้าหากไม่ได้ จำเป็นต้องส่งสัญญาณ Pslipp ไปบอกหน่วยอื่นๆให้ทราบ

3) Cache read stage

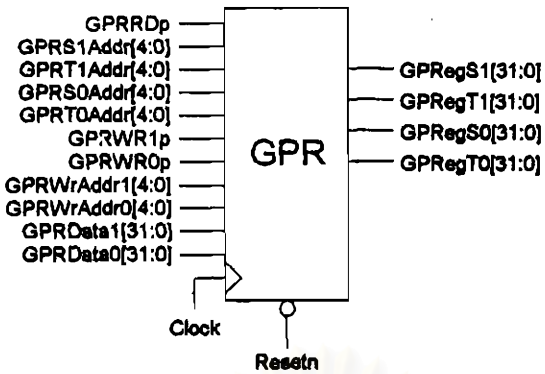
ไม่มีการประมวลผลใดใน Stage นี้ เพียงแค่ทำการรับค่าที่ได้จากการ Load จากหน่วย LSU ทางสัญญาณ LSCRD[31:0] รวมทั้งสัญญาณอื่นๆ ที่จำเป็น

4) Write back stage

ใน Stage นี้จะมีการเตรียมสัญญาณและข้อมูลที่จะเขียนลงในรีจีสเตอร์(GPR) ในช่วงขอบขาขึ้นของสัญญาณนาฬิกา พร้อมทั้งได้ตรวจสอบด้วยว่าข้อมูลที่ได้รับมาจาก LSU เป็นค่าที่ถูกต้องที่จะเขียนลงรีจีสเตอร์หรือไม่ อย่างเช่นในกรณีที่เกิด Load Miss ขึ้น (มาจากสัญญาณ LsuMissp) ข้อมูลที่ได้รับจากหน่วย LSU ทางสัญญาณ LSCRD นั้น จะไม่ถูกเขียนลงรีจีสเตอร์จนกว่าจะได้ข้อมูลที่แท้จริง ซึ่งต้องรอให้หน่วย LSU ส่งสัญญาณขอแก้ไข (LsuFixUpp) มาบอกจึงจะเขียนค่าลงในรีจีสเตอร์ได้ ทั้งนี้ในช่วงที่เกิด load miss นั้น จำเป็นต้องมีบัฟเฟอร์สำหรับเก็บตำแหน่งของรีจีสเตอร์ที่ต้องการเขียนเอาไว้ด้วย

3.1.3 GPR (General Purpose Registers)

หน่วย GPR ทำหน้าที่เก็บรีจีสเตอร์ภายใน 32 รีจีสเตอร์ ได้แก่ R0 ถึง R31 (เฉพาะที่เป็น integer) แต่ละรีจีสเตอร์มีขนาด 32 บิต โดยในหน่วย GPR นี้มีพอร์ตการอ่าน 4 พอร์ต และมีพอร์ตการเขียน 2 พอร์ต พอร์ตทั้งหมดสามารถทำการอ่านเขียนได้พร้อมกัน โดยถ้าหากมีการอ่านและเขียนรีจีสเตอร์ในตำแหน่งเดียวกันพร้อมกัน การอ่านจะได้ค่าเก่าก่อนที่จะมีการเขียน สัญญาณที่เข้าและออกในหน่วย GPR มีดังรูปที่ 3.4



รูปที่ 3.4 สัญญาณที่ติดต่อกับ GPR

- สัญญาณเข้า

GPRRDp เป็นสัญญาณร้องขอการอ่านทั้ง 4 พอร์ตที่ตำแหน่ง *GPRS1Addr[4:0]*, *GPRT1Addr[4:0]*, *GPRS0Addr[4:0]*, *GPRT0Addr[4:0]*

GPRWR0p เป็นสัญญาณร้องขอการเขียนของพอร์ตที่ 0 ที่ตำแหน่ง *GPRWrAddr0[4:0]* โดยข้อมูลที่ต้องการเขียนคือ *GPRData0[31:0]*

GPRWR1p เป็นสัญญาณร้องขอการเขียนของพอร์ตที่ 1 ที่ตำแหน่ง *GPRWrAddr1[4:0]* โดยข้อมูลที่ต้องการเขียนคือ *GPRData1[31:0]*

- สัญญาณออก

GPRegS1, *GPRegT1*, *GPRegS0*, *GPRegT0* เป็นข้อมูลที่ได้จากการอ่านรีจีสเตอร์ทั้ง 4 พอร์ต

3.1.4 SCBD (Scoreboard)

SCBD ทำหน้าที่ตรวจสอบว่ารีจีสเตอร์ที่ต้องการอ่านมีความถูกต้องของข้อมูลพร้อมที่จะให้อ่านได้หรือไม่ ดังตัวอย่างในรูปที่ 3.5 จะเห็นได้ว่าค่ารีจีสเตอร์ R10 ในคำสั่ง LW (Load Word) ยังทำงานไม่เสร็จสิ้น คำสั่ง ADD ที่ตามมาต้องการนำค่ารีจีสเตอร์ R10 ไปใช้ในการบวก ดังนั้นเพื่อป้องกันการประมวลผิดพลาด จึงจำเป็นต้องแทรกคำสั่ง NOP (No operation) เข้าไประหว่างคำสั่งที่จะทำให้เกิด Data Hazard ขึ้น ทั้งนี้ต้องพึ่งความสามารถของ compiler แต่ในงานวิจัยนี้ได้ออกแบบหน่วย SCBD เพื่อตรวจสอบเหตุการณ์ดังกล่าว แล้วส่งสัญญาณบอกให้หน่วย ISU แทรกคำสั่ง NOP

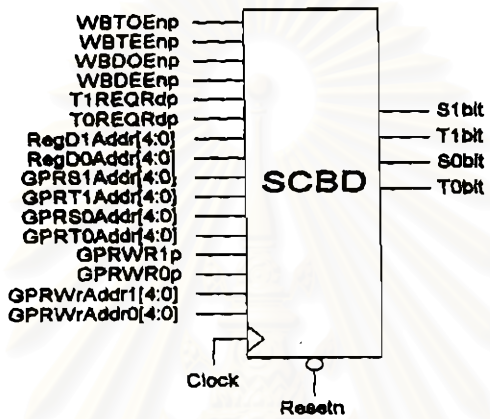
```
LW    R10, B800(R0)
ADD   R12, R11, R10
```

รูปที่ 3.5 แสดงตัวอย่างการเกิด Data Hazard

ภายในหน่วย SCBD จะมีตารางเก็บสถานะของรีจีสเตอร์ทั้ง 32 ตัวเอาไว้ เมื่อรีจีสเตอร์ตัวใดมีสัญญาณบ่งมาว่าต้องมีการเขียนค่าในอนาคต ค่าสถานะของรีจีสเตอร์ตัวนั้นจะถูกตั้งค่าเป็น '1' และจะถูก

เปลี่ยนกลับเป็น '0' เมื่อมีการเขียนค่าลงรีจีสเตอร์เรียบร้อยแล้ว ดังนั้นในช่วงเวลาที่ค่าสถานะของรีจีสเตอร์ตัวนั้นเป็น '1' อยู่ หากมีคำสั่งใดต้องการอ่าน ก็จะพบว่าไม่สามารถอ่านได้ หน่วย SCBD จะแก้ปัญหานี้ด้วยการส่งสัญญาณไปบอกหน่วย ISU จากนั้น ISU จะส่งสัญญาณ Pslipp='1' ไปบอกหน่วยประมวลผลอื่นๆ ให้ทราบ ส่งผลให้การทำงานของ Pipeline ในส่วน R Stage และ X Stage นิ่ง แต่ Pipeline ที่ C Stage และ W Stage ยังคงทำงานต่อไป จนกว่าจะมีการตั้งค่าในตารางเป็น '0' แล้ว ISU ส่งสัญญาณ Pslipp='0' แล้ว Pipeline ก็จะสามารถทำงานได้ตามปรกติ

สัญญาณที่เข้าและออกหน่วย SCBD เป็นดังรูป 3.6



รูปที่ 3.6 สัญญาณที่ติดต่อกับหน่วย SCBD

โดยมีวิธีการตั้งค่าสถานะดังนี้

- ถ้า WBTOEnp มีค่าเป็น '1' ค่าในตารางตำแหน่ง GPRT1Addr จะต้องถูกตั้งค่าเป็น '1'
- ถ้า WBTEEnp มีค่าเป็น '1' ค่าในตารางตำแหน่ง GPRT0Addr จะต้องถูกตั้งค่าเป็น '1'
- ถ้า WBDOEnp มีค่าเป็น '1' ค่าในตารางตำแหน่ง RegD1Addr จะต้องถูกตั้งค่าเป็น '1'
- ถ้า WBDEEnp มีค่าเป็น '1' ค่าในตารางตำแหน่ง RegD0Addr จะต้องถูกตั้งค่าเป็น '1'
- ถ้า GPRWR1p มีค่าเป็น '1' ค่าในตารางตำแหน่ง GPRWrAddr1 จะต้องถูกตั้งค่าเป็น '0'
- ถ้า GPRWR0p มีค่าเป็น '1' ค่าในตารางตำแหน่ง GPRWrAddr0 จะต้องถูกตั้งค่าเป็น '0'
- S1bit ได้จากการอ่านค่าในตารางตำแหน่ง GPRS1Addr
- T1bit ได้จากการอ่านค่าในตารางตำแหน่ง GPRT1Addr
- S0bit ได้จากการอ่านค่าในตารางตำแหน่ง GPRS0Addr
- T0bit ได้จากการอ่านค่าในตารางตำแหน่ง GPRT0Addr

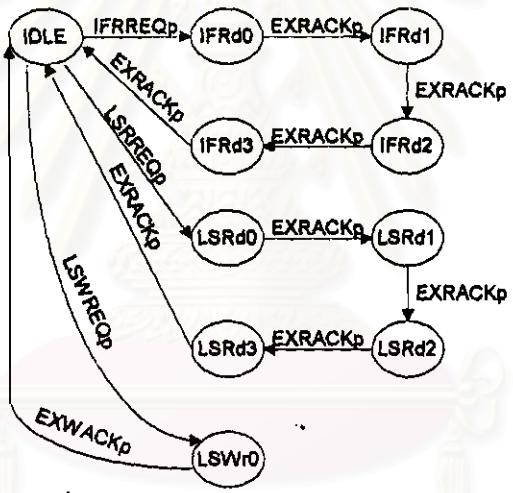
```
LW    R10, B800(R0)
LW    R10, A000(R0)
ADD   R12, R11, R10
```

รูปที่ 3.7 แสดงตัวอย่างการเกิด Data Hazard ขณะทดสอบ

ถ้าในช่วงทำการตรวจสอบความถูกต้องของการทำงานของวงจรรวมมีเหตุการณ์ดังรูป 3.7 เกิดขึ้น ซึ่งในความเป็นจริงไม่มีทางเกิดเหตุการณ์เช่นนี้ได้ เหตุการณ์ดังกล่าวเกิดเนื่องมาจากการสุ่มคำสั่งและตำแหน่งของรีจีสเตอร์ขึ้นมา เป็นผลทำให้ Output Vector ที่ได้จากโปรแกรม Verilog และภาษา C ไม่ตรงกัน จึงได้เพิ่มตารางขึ้นมาอีก 1 ตาราง ขนาดเท่ากับตารางที่กล่าวมาแล้ว โดยการทำงานมีลักษณะคล้าย Stack การตั้งค่าในตารางมีการตรวจสอบในลักษณะเดิม แต่มีการตรวจสอบเพิ่มเติมว่า เมื่อไรที่ต้องตั้งค่าในตารางเป็น '1' แล้วค่าในตารางเป็น '1' อยู่ก่อนแล้ว ก็จะไปตั้งค่าในตารางที่เพิ่มขึ้นมาให้เป็น '1' เป็นผลทำให้ต้องมีการเขียนกลับลงในรีจีสเตอร์ตำแหน่งนั้น 2 ครั้งก่อน รีจีสเตอร์ตำแหน่งนั้นจึงจะเก็บค่าที่ถูกต้องไว้

3.1.5 BIU (Bus Interface Unit)

BIU ทำหน้าที่เป็นตัวกลางในการติดต่อกับหน่วยความจำ ภายในมีองค์ประกอบหลักเป็น State Machine เพื่อใช้บอกสถานะขณะนั้นว่าควรจะทำงานใดก่อน โดย state machine ภายในหน่วย BIU เป็นดังรูปที่ 3.8

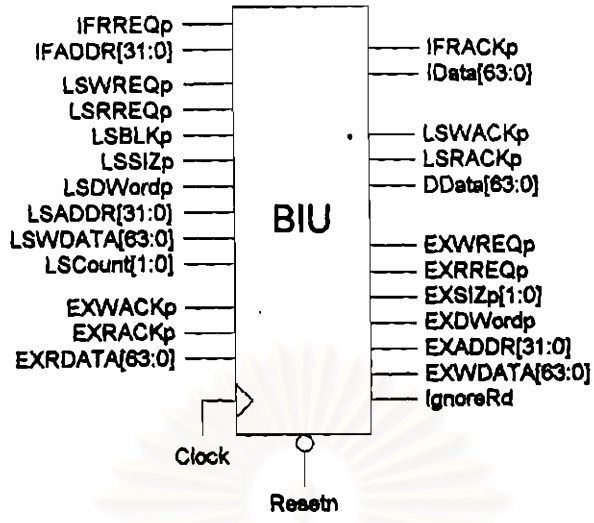


รูปที่ 3.8 state machine ภายในหน่วย BIU

ในตอนเริ่มต้น Stage อยู่ที่ iDLE จะไม่มีการติดต่อกับหน่วยความจำ ต่อมาเมื่อมีการร้องขอการอ่านหรือเขียนข้อมูลกับหน่วยความจำ สัญญาณ IFRREQp, LSRREQp หรือ LSWREQp จะเปลี่ยน Stage ของ BIU โดยพิจารณาตามความสำคัญของการติดต่อกับหน่วยความจำ โดยหน่วย BIU นี้ให้ความสำคัญกับการร้องขอการอ่านจากหน่วย IFU มากที่สุด (สัญญาณ IFRREQp) รองลงมาคือการขออ่านจากหน่วย LSU (สัญญาณ LSRREQp) และความสำคัญต่ำที่สุดคือการร้องขอการเขียนจากหน่วย LSU (สัญญาณ LSWREQp)

การอ่านข้อมูลจากหน่วยความจำ โดยปรกติแล้วจะอ่านคราวละ 4 doublewords ระหว่างที่อ่านอยู่จะไม่มีงานอื่นเข้ามาขอแทรกได้ งานวิจัยนี้ได้ออกแบบเผื่อเอาไว้ ในกรณีที่ประมวลผลโดยไม่มีแคชด้วย ในกรณีดังกล่าว หน่วย BIU นี้จะอนุญาตให้หน่วย LSU อ่านคราวละ 1 doubleword สำหรับการเขียนจะเขียนคราวละ 1 doubleword เท่านั้น

สัญญาณที่ติดต่อกับหน่วย BIU เป็นดังรูปที่ 3.9



รูปที่ 3.9 สัญญาณที่ติดต่อกับหน่วย BIU

- สัญญาณเข้า

IFRREQp ทำการร้องขอการอ่านข้อมูลจากหน่วยความจำที่ตำแหน่ง *IFADDR[31:0]* โดยร้องขอมาจากหน่วย IFU

LSRREQp ทำการร้องขอการอ่านข้อมูลจากหน่วยความจำที่ตำแหน่ง *LSADDR[31:0]* โดยร้องขอมาจากหน่วย LSU

LSWREQp ร้องขอการเขียนข้อมูลที่ตำแหน่ง *LSADDR[31:0]* ซึ่งขนาดที่จะเขียนถูกระบุโดยสัญญาณ *LSDWordp* (เขียนเป็น doubleword) หรือ *LSSIZp* (ระบุจำนวนไบต์ที่จะเขียน) โดยข้อมูลที่ต้องการเขียนคือ *LSWDATA[63:0]*

LSBLKp เป็นสัญญาณที่หน่วย LSU ร้องขอการอ่านข้อมูลคราวละ 4 doublewords

LSCount[1:0] เป็นจำนวนที่บอกการขอเขียนข้อมูลลงหน่วยความจำ

EXWACKp เป็นสัญญาณตอบรับการเขียนจากหน่วยความจำ

EXRACKp เป็นสัญญาณตอบรับการอ่าน โดยข้อมูลที่อ่านได้คือ *EXRDATA[63:0]*

- สัญญาณออก

IFRACKp สัญญาณตอบรับการอ่านข้อมูลของหน่วย IFU พร้อมทั้งข้อมูลที่อ่านได้คือ *Idata[63:0]*

LSRACKp สัญญาณตอบรับการอ่านข้อมูลของหน่วย LSU พร้อมทั้งข้อมูลที่อ่านได้คือ *Ddata[63:0]*

LSWACKp สัญญาณตอบรับการเขียนให้กับหน่วย LSU

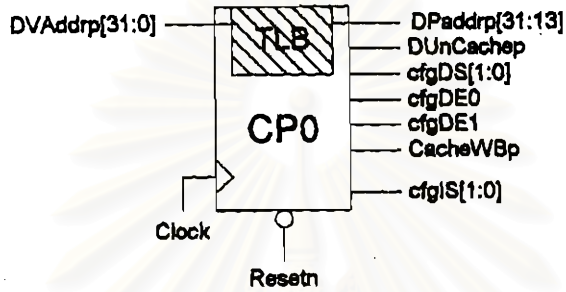
EXRREQp สัญญาณร้องขอการอ่านข้อมูลจากหน่วยความจำที่ตำแหน่ง *EXADDR[31:0]*

EXWREQp สัญญาณร้องขอการเขียนข้อมูลลงหน่วยความจำที่ตำแหน่ง *EXADDR[31:0]* โดยระบุขนาดการเขียนทาง *EXDWordp* (เขียนเป็น doubleword) หรือ *EXSIZp* (จำนวนไบต์) โดยข้อมูลที่ต้องการเขียนคือ *EXWDATA[63:0]*

3.1.6 CP0 (Coprocessor 0)

ในงานวิจัยนี้ CP0 ทำหน้าที่เป็นเพียงตัวเก็บพารามิเตอร์ที่ใช้ในการบอกขนาดของแคช, การจัดแคช และโหมดการเขียนแคชเท่านั้น สัญญาณออกทั้งหมดจึงเป็นเพียงค่าจากตัวแปรที่เก็บไว้ ส่วนในเรื่องของการเปลี่ยนค่า Virtual Address ให้เป็น Physical Address ต้องใช้ TLB(Translation Lookaside Buffer) เป็นตัวจัดการ แต่ในงานวิจัยนี้กำหนดให้ TLB เป็นกล่องดำไป ดังนั้นค่า Physical Address ที่ได้จึงเท่ากับค่า Virtual Address

สัญญาณเข้าและออกหน่วย CP0 เป็นดังรูปที่ 3.10



รูปที่ 3.10 สัญญาณที่ติดต่อกับหน่วย CP0

- สัญญาณเข้า

DVAddrp[31:0] เป็นค่า Virtual Address ที่คำนวณได้ที่ X stage

- สัญญาณออก

Dpaddrp[31:13] เป็นค่า Physical Address ที่ใช้ในการติดต่อกับหน่วยความจำ

DUnCachep มีแคช (0) / ไม่มีแคช (1)

cfgDS[1:0] ใช้ระบุขนาดของ Data Cache ที่ใช้ ได้แก่ 1K(00) / 2K(01) / 4K(10) / 8K(11)

cfgDE0 D-Cache set 0 ถูกใช้ (1) / ไม่ถูกใช้ (0)

cfgDE1 D-Cache set 1 ถูกใช้ (1) / ไม่ถูกใช้ (0)

CacheWBp ตั้งโหมดการเขียนแคช Write Back(1)/ Write Through (0)

3.1.6 MEM (memory)

เป็นการจำลองหน่วยความจำขึ้นมาสำหรับใช้ในการทดสอบ โดยขนาดของหน่วยความจำที่กำหนดจะเปลี่ยนแปลงตามขนาดของโปรแกรม Benchmark ที่ใช้ในการทดสอบ โดยสามารถจองได้ขนาดใหญ่ที่สุด 16,777,216 คำสั่ง โดยขึ้นอยู่กับขนาด RAM ของเครื่อง เครื่องคอมพิวเตอร์ที่ใช้ในงานวิจัยนี้มี RAM ขนาด 256 MB

สัญญาณควบคุมการทำงานดังรูปที่ 3.11

- สัญญาณเข้า

EXWREQp คือสัญญาณร้องขอการเขียนข้อมูลลงหน่วยความจำ

EXRREQp คือสัญญาณร้องขอการอ่านข้อมูลจากหน่วยความจำ

EXSIZp[1:0] ใช้ระบุขนาดของการเขียนข้อมูล ได้แก่ 1 byte, 1 halfword (2 bytes), 3 bytes และ 1 word (4 bytes) โดย EXSIZp จะมีค่าเท่ากับ 00, 01, 10 และ 11 ตามลำดับขนาดของข้อมูลที่กล่าวไว้

EXDWordp เป็นการขอเขียนข้อมูลเป็น doubleword

EXADDR[31:0] คือตำแหน่งที่ต้องการติดต่อกับหน่วยความจำ

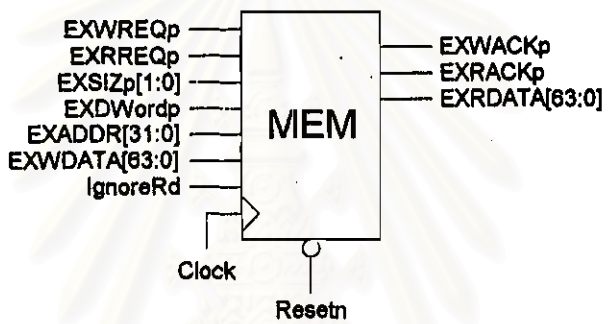
EXDataIn[63:0] คือข้อมูลขนาด 64 บิตที่ต้องการเขียนลงหน่วยความจำ

- สัญญาณออก

EXWACKp เป็นสัญญาณตอบรับการเขียน

EXRACKp เป็นสัญญาณตอบรับการอ่าน

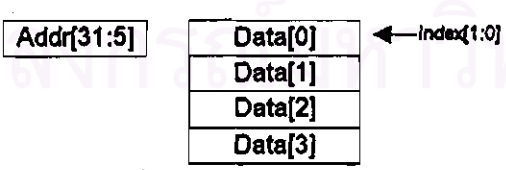
EXDateOut[63:0] เป็นข้อมูลที่ได้จากการอ่าน



รูปที่ 3.11 สัญญาณที่ติดต่อกับหน่วย MEM

3.1.8 WBB (Write Back Buffer)

หน่วยนี้จะถูกใช้งานเมื่อเป็นโหมดการเขียนแบบ write back เท่านั้นกล่าวคือถ้าเป็นโหมด write through ก็สามารถตัดหน่วย WBB ออกไปได้ โดยไม่มีผลกระทบต่อการประมวลผลแต่อย่างใด ภายในหน่วย WBB มีบัฟเฟอร์ขนาด 4 doublewords (ตามขนาด block size ของ D-Cache data) และบัฟเฟอร์สำหรับเก็บค่าแอดเดรสขนาด 27 บิตดังรูปที่ 3.12



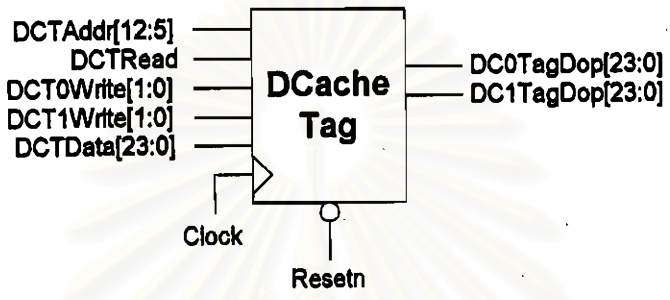
รูปที่ 3.12 บัฟเฟอร์ภายในหน่วย WBB

เมื่อเกิด Load Miss ขึ้น และข้อมูลที่อยู่ใน D-Cache บรรทัดนั้นมีการเปลี่ยนแปลง (dirty bit = '1') ซึ่งหน่วย LMC (Load Miss Control) ตรวจสอบเหตุการณ์ดังกล่าวได้จาก State Machine ของ LMC เท่ากับ WBPRO (จะกล่าวถึง State Machine ของ LMC ในหน่วย LMC ภายใน LSU) หน่วย WBB จะทำการเก็บค่า WBBAddrIn ที่เข้ามาลงในบัฟเฟอร์ จากนั้นจะอ่านข้อมูลของ D-Cache Data ในบรรทัดที่ต้องการเขียนลง

หน่วยความจำ และส่งค่าสัญญาณ WBsy เป็น '1' เพื่อบอกให้รู้ว่าบัฟเฟอร์ไม่ว่างแล้ว และเมื่อเก็บข้อมูลจาก D-Cache Data ครบทั้ง 4 doublewords แล้ว หน่วย WBB จะส่งสัญญาณ WBDone เป็น '1'

3.1.9 D-Cache Tag

D-Cache Tag ทำหน้าที่เก็บ Tag และ Status bits ได้แก่ Valid bit, Write Back (WB) bit เพื่อใช้ในการตรวจสอบการอ้างอิงตำแหน่งของหน่วยความจำ โดยสัญญาณที่ใช้ติดต่อกับ D-Cache Tag เป็นดังรูปที่ 3.13



รูปที่ 3.13 สัญญาณที่ติดต่อกับ D-Cache Tag

- สัญญาณเข้า

DCTRead เป็นสัญญาณที่ LSU ร้องขอการอ่านข้อมูลในตำแหน่ง *DCTAddr[12:5]*

DCT0Write[1:0] เป็นสัญญาณที่ LSU ร้องขอการเขียนข้อมูลพอร์ต 0 ที่ตำแหน่ง *DCTAddr[12:5]* โดย DCT0Write[1] เป็นการขอตั้งค่า WB bit และ DCT0Write[0] เป็นการขอเขียนข้อมูล *DCTData[23:0]*

DCT1Write[1:0] เป็นสัญญาณที่ LSU ร้องขอการเขียนข้อมูลพอร์ต 1 ที่ตำแหน่ง *DCTAddr[12:5]* โดย DCT1Write[1] เป็นการขอตั้งค่า WB bit และ DCT1Write[0] เป็นการขอเขียนข้อมูล *DCTData[23:0]*

- สัญญาณออก

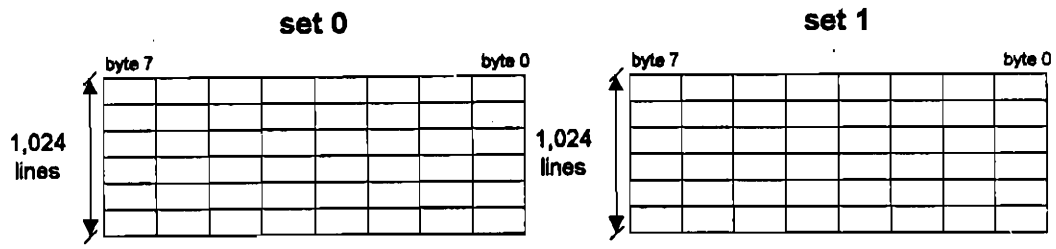
DC0TagDop[23:0] เป็นข้อมูลที่ส่งไปให้ LSU ซึ่งอ่านได้จาก D-Cache Tag set 0 โดย DC0TagDop[23:2] เป็นค่า Tag, DC0TagDop[1] เป็นค่า Valid bit และ DC0TagDop[0] เป็นค่า WB bit

DC1TagDop[23:0] เป็นข้อมูลที่ส่งไปให้ LSU ซึ่งอ่านได้จาก D-Cache Tag set 1 โดย DC1TagDop[23:2] เป็นค่า Tag, DC1TagDop[1] เป็นค่า Valid bit และ DC1TagDop[0] เป็นค่า WB bit

3.1.10 D-Cache Data

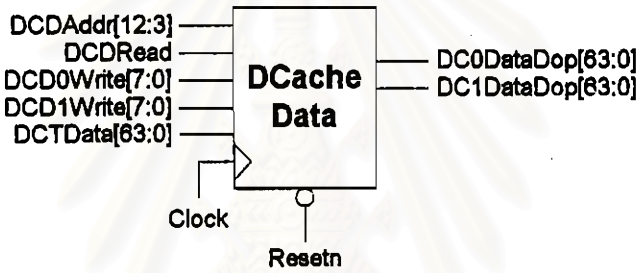
D-Cache Data ทำหน้าที่เป็นหน่วยความจำที่อยู่ระหว่างหน่วยประมวลผลกลางกับหน่วยความจำหลัก เพื่อเพิ่มความเร็วในการติดต่อกับหน่วยความจำ

ภายใน D-Cache Data ถูกออกแบบให้เป็นแคชชนิด 2-way set associative แต่ละ set มีขนาด 8 KBytes ซึ่งเป็นขนาดของการจัดแคชที่ใหญ่ที่สุดที่ใช้ในงานวิจัย ลักษณะของ D-Cache Data ถูกแสดงดังรูปที่



รูปที่ 3.14 แสดงการออกแบบแคชในหน่วย D-Cache Data

3.14 ส่วนการใช้งานจริง ขึ้นอยู่กับการกำหนดขนาดของแคชในหน่วย CP0 ดังที่ได้กล่าวมาแล้ว โดยสัญญาณที่ใช้ในการติดต่อ D-Cache Data เป็นดังรูปที่ 3.15



รูปที่ 3.15 สัญญาณที่ติดต่อกับ D-Cache Data

- สัญญาณเข้า

DCDRead เป็นสัญญาณที่ LSU ร้องขอการอ่านข้อมูลในตำแหน่ง *DCDAddr[12:3]*

DCD0Write[7:0] เป็นสัญญาณที่ LSU ร้องขอเขียนข้อมูลในพอร์ต 0 โดยสัญญาณทั้ง 8 บิตคือการขอเขียนข้อมูลในแต่ละไบต์ และข้อมูลที่ต้องการเขียนลง D-Cache Data คือ *DCDData[63:0]*

DCD1Write[7:0] เป็นสัญญาณที่ LSU ร้องขอเขียนข้อมูลในพอร์ต 1 โดยสัญญาณทั้ง 8 บิตคือการขอเขียนข้อมูลในแต่ละไบต์ และข้อมูลที่ต้องการเขียนลง D-Cache Data คือ *DCDData[63:0]*

- สัญญาณออก

DC0DataDop[63:0] เป็นข้อมูลที่ส่งไปให้ LSU ซึ่งอ่านได้จาก D-Cache Data Set 0

DC1DataDop[63:0] เป็นข้อมูลที่ส่งไปให้ LSU ซึ่งอ่านได้จาก D-Cache Data Set 1

3.1.11 TEST

หน่วย TEST ทำหน้าที่นับจำนวนเหตุการณ์ที่เกิดขึ้นทั้งหมด ซึ่งได้แก่ Load Hit, Load Miss, Store Hit, Sore Miss และเหตุการณ์ "STALL" Pipeline แบบต่างๆ ภายในประกอบไปด้วย Counter ที่ใช้ในการนับเพิ่มที่ละ 1 ตามเหตุการณ์ที่เกิดขึ้นจริง หน่วย TEST ถูกใช้ในการวัดสมรรถนะของหน่วย LSU ที่ออกแบบ

3.1.12 LSU (Load / Store Unit)

LSU ทำหน้าที่ประมวลผลคำสั่ง Load และคำสั่ง Store ซึ่งเป็นคำสั่งที่ใช้ในการติดต่อกับหน่วยความจำภายในหน่วย LSU ทำงานเป็น 4-stage pipeline รายละเอียดของหน่วย LSU จะถูกกล่าวถึงในบทต่อไป

3.2 การทำงานของไมโครโพรเซสเซอร์

ไมโครโพรเซสเซอร์ที่ออกแบบทำงานเป็น Pipeline Stage ดังรูปที่ 3.16 Stage การทำงานถูกแบ่งออกเป็น 5 stage ได้แก่ Instruction Fetch Stage, Read Register File and Instruction Decoding Stage, Execution Stage, Cache Read Stage และ Write Back Stage โดยแต่ละ Stage มีหน้าที่การทำงานดังนี้

1. Instruction Fetch

ใน Stage นี้ หน่วย IFU ทำการคำนวณค่า Program Counter (PC) เพื่อดึงคำสั่งที่ต้องประมวลผลในขณะนั้น โดยในช่วงขอบขาขึ้นของสัญญาณนาฬิกาจะดึงคำสั่งมาจากหน่วยความจำ

2. Read Register File and Instruction Decoding

ใน Stage นี้ หน่วย ISU จะทำการถอดรหัสคำสั่งเพื่อพิจารณาว่าคำสั่งนั้นๆ ควรจะส่งไปให้หน่วยใดประมวลผลต่อ โดยในงานวิจัยนี้จะเลือกเฉพาะคำสั่งที่ส่งไปให้หน่วย LSU และในช่วงขอบขาขึ้นของสัญญาณนาฬิกาจะอ่านข้อมูลจาก General Purpose Registers (GPR)

3. eXecution

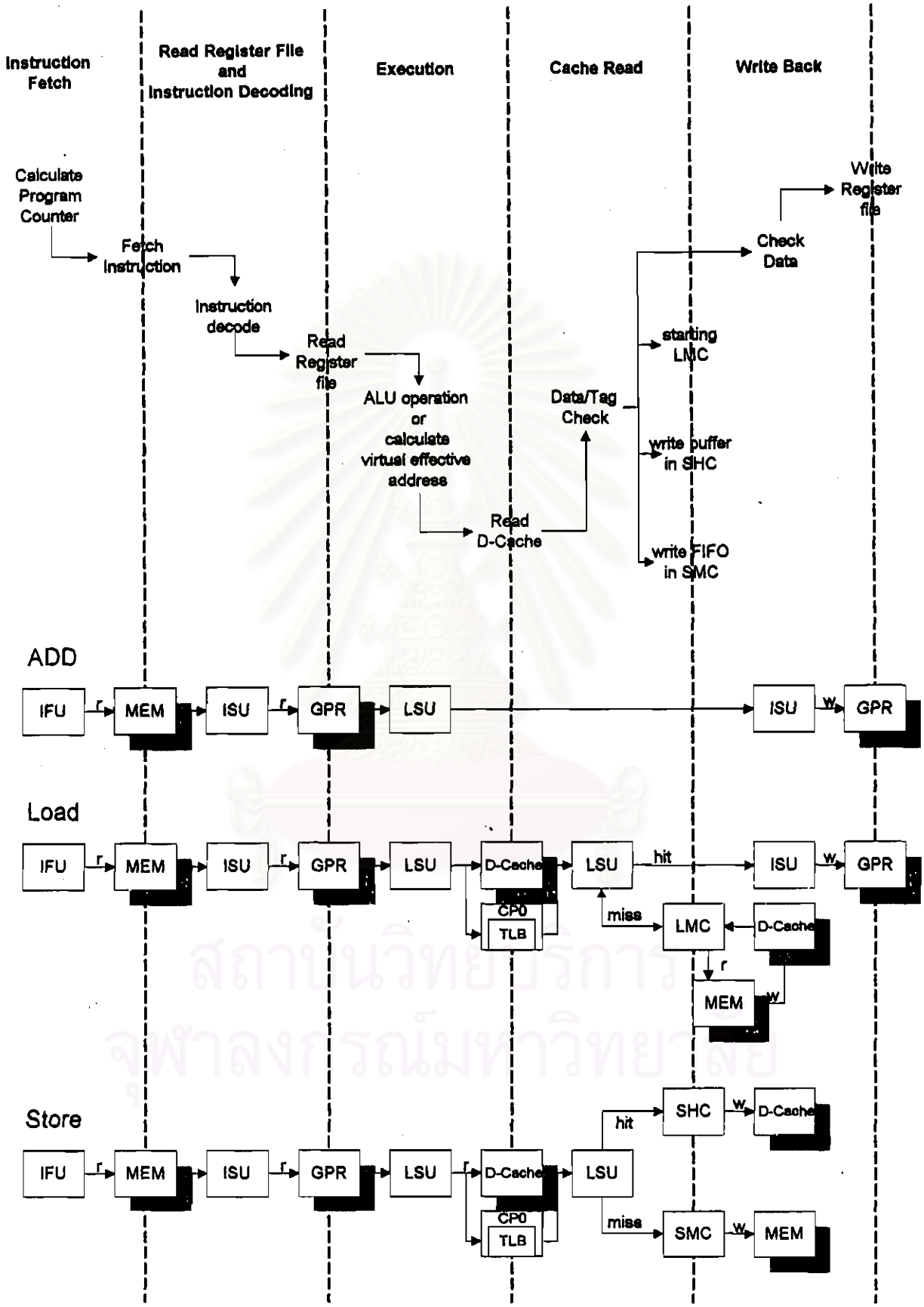
ใน Stage นี้ หน่วย LSU จะทำการประมวลผลคำสั่ง ADD หรือคำนวณหาค่า Virtual Effective Address เพื่อหาค่าแอดเดรสที่ต้องการ Load หรือ Store และในช่วงขอบขาขึ้นของสัญญาณนาฬิกาจะนำค่าแอดเดรสดังกล่าวใช้เป็นตำแหน่งในการชี้ D-Cache ซึ่งในระหว่างนั้นก็ส่งค่า Virtual Effective Address ไปให้หน่วย CP0 เพื่อให้ TLB แปลงเป็นค่า Physical Address

4. Cache read

ใน Stage นี้ หน่วย LSU จะพิจารณาข้อมูลที่อ่านได้จากแคช เพื่อตรวจสอบดูว่าคำสั่ง Load หรือ Store ที่กำลังประมวลผลอยู่นั้นเกิด Hit หรือ Miss และในช่วงขอบขาขึ้นของสัญญาณนาฬิกา หน่วย LMC จะเริ่มทำงานในกรณีที่เกิด Load Miss หรือหน่วย SHC จะเก็บข้อมูลลงบัฟเฟอร์ในกรณีที่เกิด Store Hit หรือหน่วย SMC เก็บข้อมูลลง FIFO ในกรณีที่เกิด Store Miss

5. Write back

ใน stage นี้หน่วย ISU จะทำการเตรียมข้อมูลที่ต้องการเขียนลงรีจีสเตอร์ หลังจากนั้นจะเขียนข้อมูลดังกล่าวลงรีจีสเตอร์ในช่วงขอบขาขึ้นของสัญญาณนาฬิกา



รูปที่ 3.17 ขั้นตอนการประมวลผลของไมโครโพรเซสเซอร์ที่ออกแบบ

ผู้วิจัยได้เน้นการออกแบบเฉพาะส่วน LSU ดังนั้นไมโครโพรเซสเซอร์ที่ออกแบบจึงสามารถประมวลผลได้เฉพาะคำสั่งที่ทำงานใน LSU เท่านั้น ซึ่งได้แก่

1. คำสั่ง ADD
2. ชุดคำสั่ง LOAD ซึ่งประกอบไปด้วย LB (Load Byte), LBU (Load Byte Unsigned), LH (Load Halfword), LHU (Load Halfword Unsigned), LW (Load Word), LWL (Load Word Left) และ LWR (Load Word Right)
3. ชุดคำสั่ง STORE ซึ่งประกอบด้วย SB (Store Byte), SH (Store Halfword), SW (Store Word), SWL (Store Word Left) และ SWR (Store Word Right)

โดยแต่ละคำสั่งมีกระบวนการในการประมวลผลดังต่อไปนี้

- กระบวนการในการประมวลผลคำสั่ง ADD

การประมวลผลคำสั่ง ADD เป็นไปตามรูปที่ 3.17 โดย Block ที่ไม่มีเงาคือหน่วยประมวลผลที่ทำงานคำสั่ง ADD ในแต่ละ Stage ของ Pipeline ส่วน Block ที่มีเงาคือหน่วยความจำที่ใช้ (ได้แก่ GPR, D-Cache และ MEM) การประมวลผลของคำสั่ง ADD เริ่มต้นที่หน่วย IFU อ่านข้อมูลที่เป็นคำสั่งมาจากหน่วยความจำ จากนั้นส่งคำสั่งไปให้หน่วย ISU เพื่อถอดรหัส เมื่อพบว่าเป็นคำสั่ง ADD จะส่งตำแหน่งของรีจิสเตอร์ที่ต้องใช้ไปให้หน่วย GPR เพื่อขออ่าน ต่อมาการบวกจะทำที่หน่วย LSU ในช่วง Execution Stage แล้วหน่วย ISU จะเริ่มเตรียมข้อมูลที่ได้จากการ ADD เขียนค่าลงในรีจิสเตอร์ในช่วง Write Back Stage

- กระบวนการในการประมวลผลคำสั่ง Load

การประมวลผลคำสั่ง Load เป็นไปดังรูปที่ 3.17 โดยในช่วงแรกจะประมวลผลเหมือนกันคำสั่ง ADD และจะต่างกันเมื่ออยู่ใน Execution Stage หน่วย LSU จะทำการคำนวณค่า Virtual Effective Address แล้วทำการขออ่านข้อมูลจาก D-Cache หลังจากนั้นในช่วง Cache Read Stage หน่วย LSU จะพิจารณาดูว่าค่าที่ได้จากการอ่านแคชนั้น Hit หรือ Miss ถ้าหากว่าผลการแคชเป็น Hit การประมวลผลคำสั่ง Load ที่ Pipeline C Stage จะส่งค่าที่ได้จากการโหลดไปให้ ISU เพื่อเตรียมเขียนลงในรีจิสเตอร์ต่อไป แต่ถ้าหากการอ่านแคชเกิดเป็น Miss การประมวลผลคำสั่ง Load ที่ Pipeline C Stage จะส่งการประมวลผลไปให้หน่วย LMC จัดการต่อ โดยหน่วย LMC ร้องขอข้อมูลจากหน่วยความจำเพื่อเขียนเก็บไว้ในแคช แล้วจึงนำข้อมูลที่ได้จากการโหลดส่งไปให้หน่วย ISU ต่อไป

- กระบวนการในการประมวลผลคำสั่ง Store

การประมวลผลคำสั่ง Store เป็นไปดังรูปที่ 3.17 โดยในช่วงแรกจะประมวลผลเหมือนกับคำสั่ง Load จนกระทั่งถึง Cache Read Stage จะพิจารณาว่าการติดต่อกับแคชนั้น Hit หรือ Miss ซึ่งถ้าหากผลเป็น Hit การประมวลผลคำสั่ง Store ที่ Pipeline C Stage จะส่งข้อมูลที่ต้องการเก็บไปให้หน่วยประมวลย่อย SHC ภายใน LSU เพื่อพักไว้ในบัฟเฟอร์ แล้วรอเขียนข้อมูลลงแคชต่อไปในอนาคต แต่ถ้าผลการติดต่อกับแคชเกิดเป็น Miss ประมวลผลคำสั่ง Store ที่ Pipeline C Stage จะส่งข้อมูลไปให้หน่วยประมวลผลย่อย SMC ภายใน LSU เพื่อพักข้อมูลไว้ที่ FIFO รอการเขียนข้อมูลลงหน่วยความจำต่อไป

3.3 สรุป

ในบทนี้ได้กล่าวถึงหน้าที่การทำงานและวิธีการออกแบบของแต่ละหน่วยประมวลผลที่อยู่ภายในไมโครโพรเซสเซอร์ที่ถูกออกแบบ และได้บรรยายถึงขั้นตอนในการทำงานของแต่ละคำสั่ง ซึ่งไมโครโพรเซสเซอร์ที่ออกแบบนี้ทำงานได้เฉพาะคำสั่งที่ประมวลผลใน LSU เท่านั้น ในบทต่อไปจะกล่าวเน้นในส่วนของ Load / Store Unit (LSU)