Project Report CPE 224 CPE 224 2/2017: Term Project

Part 1: *Simple MIPS instruction*

Format	Instruction Name
R-type	ADD, SUB, AND, OR, SLT
I-type	LW, SW, ADDI

Summaries: เพิ่ม control unit ที่ใช้ แยกคำสั่ง ระหว่าง R-type , I-type ซึ่งเราสามารถ อ้างอิงได้ตาม mips instruction sheet ดังนี้

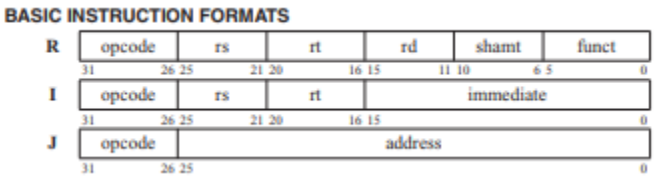

Figure 1 แสดง Basic Instruction formats

ซึ่งในส่วนของ คำสั่งประเภท R-type นั้น จะมี Op-code เป็น 6'b000000 **เหมือนกัน ต่างที่** funct **เพราะ** R type **มีการใช้** ALU operation **ที่หลากหลาย** (+ - shift and or) **จะใช้** operation **ไหนขึ้นอยู่กับ** funct **ซึ่งจะถูกควบคุมผ่าน** Alu control ซึ่งจะแตกต่างกันในส่วน ของ funct ที่ใช้ ซึ่งจะถูกควบคุม โดย ALU control เพื่อสั่งใช่งาน คำสั่ง ALU ต่อไป

Part 2: *Design an additional datapath to support branch-on-equal (BEQ) and jump (J) instructions.*

Summaries: branch on equal หรือ BEQ จะเป็นคำส่งประเภท I-type จาก mips instruction sheet ระบุว่า (**if**(R[rs]==R[rt]) **then** PC=PC+4+BranchAddr) **แสดง**ว่า คำสั่งนี้ BEQ $s,$t,offset **เมื่อ** $s == $t **เมื่อไหร่** ตัว processor จะสั่งให้ทำการ branch **โดย**สามารถควบคุมผ่าน ALU ในคำสั่งที่เป็น subtract และเพิ่ม zero bit **เข้าไป** เพื่อเช็คว่า R[rs] == R[rt] หรือ**เปล่า** ? ถ้าเท่ากันแล้ว zero bit จะเท่ากับ 1

แต่ในส่วนของคำสั่ง Jump หรือ J เป็นคำสั่งประเภท J-type ดังภาพที่ 1 ซึ่งมีรูปแบบคำสั่งเป็น J target_address (PC = Address) เป็นคำสั่งที่จะทำการ jump เลย หากทำงาน instruction นี้

Part 3: *Extend your processor to support LB and SLL instruction.*

Summaries: ในส่วนของ load-byte หรือ LB เป็นคำสั่งประเภท I-type เช่นเดียวกับ load-word โดยจะแตกต่างกันที่ load-byte เป็นการดึงข้อมูลมาที่ละ 1 แต่ load-word เป็นการดึงข้อมูลมาที่ละ 4 byte ซึ่งคำสั่งจะเป็น lb $t, offset($s) ซึ่งตาม mips reference แล้ว จะเป็นการใช้บวกที่ ALU เพื่อเป็นการ load memory จาก bype ที่เป็น base address + offset

แต่ในส่วนของ Shift-left-logical จะเป็นคำสั่งประเภท R-type ที่จะทำการ shift logical bit ไปยังทางซ้าย จาก mips instruction sheet ระบุว่า R[rd] = R[rt] << shamt ดังนั้นจะเป็นการส่งคำสั่งไปยัง ALU ผ่าน ALU control ผ่าน funct ใหม่ ที่จะทำการเลื่อน bit ไปเท่ากับ shamt

ซึ่งได้ทำการเพิ่ม input ของ shamt เข้าไปใหม่ในส่วนของ top-modul -> ALU ซึ่งจาก mips reference sheet พบว่า shamt คือ ตำแหน่ง instruction[10:6] และทำการเลื่อนบิตไปทางซ้าย โดยเขียนคำสั่งเป็น s = b << shamt

Part 4: *Extend your processor to support JAL, JR.*

Summaries: ในส่วนของ Jump-and-link หรือ JAL จะเป็น คำสั่งประเภท J-type ซึ่งตาม mips reference ระบุว่า R[31]=PC+8; PC=JumpAddr ซึ่งในส่วนของR[31] ก็คือ $ra หรือ return address ที่จะระบุไปว่าว่า ตำแหน่ง PC ต่อไปคืออะไรตามที่ ระบุไว้ แต่เราได้เปลี่ยน จาก PC+8 เป็น PC+4 เนื่องจากว่าเราได้ตัดปัญหาเรื่อง memory-stall ออกไป ซึ่งสิ่งที่เราต้องเพิ่มคือ multiplexer ตรงส่วนของ RegDst และ MemtoReg ดังภาพ เพื่อเป็นตัวแยกว่าตำแหน่งไหนเอาไว้เก็บ $ra

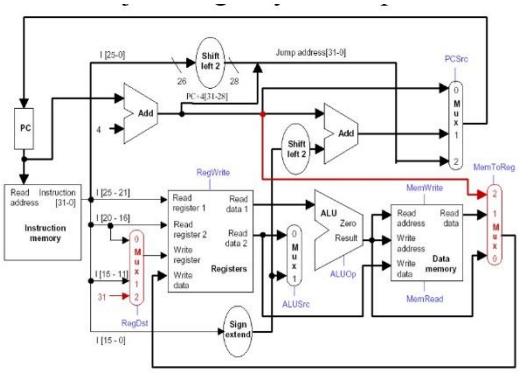

Figure 2 แสดง datapath ที่เพิ่มสายของ JAL

ส่วนคำสั่ง Jump register หรือ JR จะเป็นคำสั่งประเภท R-type ซึ่งตาม mips reference ระบุว่า PC=R[rs] ซึ่งในส่วนของ R[rs] เป็น $ra เราจะทำตัวเช็คหนึ่งตัวขึ้นมาชื่อว่า Jumregister ซึ่งถ้าคำสั่งเป็น JR เราจะระบุให้ Jumpregister เป็น 1 แต่ถ้าเป็นกรณีอื่นจะให้เป็น 0 ซึ่งตรงส่วน new_address จะแก้เป็น assign **new_address = JumpReg ? read_data_1:** ซึ่งคำสั่ง jump register จะเป็นการ jump ไปยังตำแหน่ง $ra ที่เก็บไว้

สมาชิก

59070501024, 59070501043, 59070501047, 59070501069