รายการอ้างอิง

[1] Kozyrakis, C., and D. Patterson. A new direction for computer architecture research. IEEE Computer. (1998): 24-32.

[2] Hines, S., J. Green, G. Tyson, and D. Whalley. Improving program efficiency by packing instructions into registers. In Proceedings of the 2005 ACM/IEEE International Symposium on Computer Architecture. Barcelona, Spain (2005): 260-271.

[3] Hines, S., J. Green, G. Tyson and D. Whalley, Improving Program Efficiency by Packing Instructions into Registers. ACM SIGARCH Computer Architecture News, Proceedings of the 32nd Annual International Symposium on Computer Architecture (2005): 260-271.

[4] Hines, S., G. Tyson, and D. Whalley. Reducing Instruction Fetch Cost by Packing Instructions into RegisterWindows. In Proceedings of the 38th annual ACM/IEEE International Symposium on Microarchitecture. Portland, Oregon (2005): 19-29.

[5] Maly, W. Cost of Silicon Viewed from VLSI Design Perspective. Proceedings of the 31st annual conference on Design automation. San Diego, California (1994): 135-142.

[6] Motorola, Inc., MC68HC908GP20, HCMOS Microcontroller Unit Rev 2.0, 1998.

[7] Jamil, T. RISC versus CISC. IEEE Potentials 14 (August/September 1995): 13-16.

[8] A. Beszedez, R. Ferenc, T. Gyimothy, A. Dolenc, K. Karsisto. Survey of code-size reduction methods. ACM Computing Surveys Volume 35 Issue 3 (September 2003) : 223-267.

[9] T Lindholm, F Yellin. The Java™ Virtual Machine Specification. 2nd ed. Boston : Addison-Wesley Longman Publishing Co., Inc., 1999

[10] Advanced RISC Machines Ltd., An Introduction to Thumb. Developer Technical Document, 1995.

[11] Kissell., K. D., MIPS16: High-density MIPS for the embedded market. Proceedings of Real Time Systems (1997).

[12] Krishnaswamy, A., and R. Gupta. Profile Guided Selection of ARM and Thumb Instruction. Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems. Berlin, German (2002): 56-64.

[13] O'Connor, J.M., and M. Tremblay. picoJava-I: the Java virtual machine in hardware. IEEE Micro 17 (March/April 1997): 45-53.

[14] Nanthanavoot, P., and P. Chongstitvatana. Code-Size Reduction for Embedded Systems using Bytecode Translation Unit. Conf. of Electrical/Electronics, Computer, Telecommunications, and Information Technology. Thailand (2004).

[15] Burutarchanai, A., P. Nanthanavoot, C. Aporntewan, and P. Chongstitvatana. A stack-based processor for resource efficient embedded systems. Proc. of IEEE TENCON. Thailand (2004): 439-442.

[16] Burutarchanai, A., V.Kotrajaras, and P. Chongstitvatana. A fast instruction fetch unit for an embedded stack processor. Proc. of Int. Conf. on Information and Communication Technologies. Thailand (2004).

[17] Nanthanavoot, P., A. Burutarchanai, and P. Chongstitvatana. Instruction packing for a 32-bit resource efficient processor. National Science and Technology Development Agency (NSTDA) Annual Conference. Thailand (2005).

[18] Xilinx, Inc. MicroBlaze Processor Reference Guide. San Jose, California : Xilinx User Guide, 2002.

ภาคผนวก

ภาคผนวก ก
ความหมายของคำสั่งและการให้รหัสคำสั่ง

ในส่วนนี้แสดงรายละเอียดเพิ่มจากเนื้อหาในบทที่ 3 เกี่ยวกับชุดคำสั่งของหน่วยประมวลผลนี้

ก.1 ความหมายของคำสั่ง

การอธิบายความหมายของคำสั่งแสดงในตารางที่ ก.2 ก.3 ก.4 และ ก.5 มีการใช้สัญลักษณ์ต่างๆ เพื่อความกะทัดรัดในการอธิบายการทำงานของคำสั่ง ความหมายของสัญลักษณ์ที่ใช้แสดงในตารางที่ ก.1

ตารางที่ ก.1 แสดงความหมายของสัญลักษณ์ที่ใช้ในการอธิบายการทำงานของแต่ละคำสั่ง

สัญลักษณ์	คำอธิบาย
AC	ค่าที่เก็บในรีจิสเตอร์ AC
FP	ค่าที่เก็บในรีจิสเตอร์ FP
IR	ค่าที่เก็บในรีจิสะเตอร์ IR
TEMP	ค่าที่เก็บในรีจิสเตอร์ TEMP
PC	ค่าที่เก็บในรีจิสเตอร์ PC
C	ค่าที่เก็บในรีจิสเตอร์ C
AC	ค่าที่เก็บในรีจิสเตอร์ AC
ADS	ระบุเลขที่อยู่แบบสัมบูรณ์
OFFSET	ระบุระยะของการกระโดด
DISP	ระบุระยะทางจากตัวชี้
IMM	ค่าของตัวถูกดำเนินการแบบทันที
LVAR	ระบุตัวแปรเฉพาะที่
M[ADDR]	ค่าของหน่วยความจำตำแหน่งที่ ADDR
lit8bit	ค่าที่ต้องการใส่ในรีจิสเตอร์ AC ขนาด 8 บิต
lit24bit	ค่าที่ต้องการใส่ในรีจิสเตอร์ AC ขนาด 24 บิต
X op Y, op X	ผลลัพธ์จากการคำนวณของวงจรหน่วยคำนวณและตรรกะโดยสัญญาณนำเข้าเป็นสัญญาณ X และ Y
FS	คำสั่ง CALL ขนาดแอดทิเวชันเรคคอร์ดของโปรแกรมย่อยที่เรียก คำสั่ง RET ขนาดแอดทิเวชันเรคคอร์ดของโปรแกรมย่อยที่กลับออกมา
X ← Y;	ค่าของรีจิสเตอร์ X เท่ากับสัญญาณ Y

ตารางที่ ก.1 แสดงความหมายของสัญลักษณ์ที่ใช้ในการอธิบายการทำงานของแต่ละคำสั่ง (ต่อ)

สัญลักษณ์	คำอธิบาย
{X, Y}	สัญญาณที่ได้จากการนำสัญญาณ X และ Y มารวมกัน โดยสัญญาณ X อยู่ในตำแหน่งบิตสูง และสัญญาณ Y อยู่ในตำแหน่งบิตต่ำ
X[Y:Z]	ค่าของสัญญาณ X บิต Y ถึงบิต Z

ตารางที่ ก.2 อธิบายความหมายและการทำงานของคำสั่งประเภทเลขคณิตและตรรกะ

Mnemonic	Operand	Description	Operation	C	Format
ADD	LVAR	Add content of AC by indicated local variable	AC ← AC + M[FP-LVAR];	Yes	Short
SUB	LVAR	Subtract content of AC by indicated local variable	AC ← AC – M[FP-LVAR];	Yes	Short
MUL	LVAR	Multiply content of AC by indicated local variable	AC ← AC * M[FP-LVAR];	No	Short
DIV	LVAR	Divide content of AC by indicated local variable	AC ← AC / M[FP-LVAR];	No	Short
OR	LVAR	Bitwise or content of AC by indicated local variable	AC ← AC | M[FP-LVAR];	No	Short
AND	LVAR	Bitwise and content of AC by indicated local variable	AC ← AC & M[FP-LVAR];	No	Short
XOR	LVAR	Bitwise exclusive or content of AC by indicated local variable	AC ← AC ^ M[FP-LVAR];	No	Short
MOD	LVAR	Modulo content of AC by indicated local variable	AC ← AC % M[FP-LVAR];	No	Short
ADDC	LVAR	Add content of AC by indicated local variable with carry	AC ← AC+M[FP-LVAR] + C;	Yes	Short
SUBC	LVAR	Subtract content of AC by indicated local variable with carry	AC ← AC-M[FP-LVAR] – C;	Yes	Short
EQ	LVAR	Equal comparison with indicated local variable	if (AC == M[FP-LVAR]) AC ← 1; else AC ← 0;	No	Short
NEQ	LVAR	Not equal comparison with indicated local variable	if (AC != M[FP-LVAR]) AC ← 1; else AC ← 0;	No	Short
GT	LVAR	Greater than comparison with indicated local variable	if (AC > M[FP-LVAR]) AC ← 1; else AC ← 0;	No	Short
GE	LVAR	Greater or equal comparison with indicated local variable	if (AC >= M[FP-LVAR]) AC ← 1; else AC ← 0;	No	Short
LT	LVAR	Less than comparison with indicated local variable	if (AC < M[FP-LVAR]) AC ← 1; else AC ← 0;	No	Short
LE	LVAR	Less or equal comparison with indicated local variable	if (AC <= M[FP-LVAR]) AC ← 1; else AC ← 0;	No	Short

ตารางที่ ก.2 อธิบายความหมายและการทำงานของคำสั่งประเภทเลขคณิตและตรรกะ (ต่อ)

Mnemonic	Operand	Description	Operation	C	Format
ADDI	IMM	Add content of AC by immediate value of operand	AC ← AC + IMM;	Yes	Short
SUBI	IMM	Subtract content of AC by immediate value of operand	AC ← AC – IMM;	Yes	Short
MULI	IMM	Multiply content of AC by immediate value of operand	AC ← AC * IMM;	No	Short
DIVI	IMM	Divide content of AC by immediate value of operand	AC ← AC / IMM;	No	Short
ORI	IMM	Bitwise or content of AC by immediate value of operand	AC ← AC | IMM;	No	Short
ANDI	IMM	Bitwise and content of AC by immediate value of operand	AC ← AC & IMM;	No	Short
XORI	IMM	Bitwise exclusive or content of AC by immediate value	AC ← AC ^ IMM;	No	Short
MODI	IMM	Modulo content of AC by immediate value of operand	AC ← AC % IMM;	No	Short
ADDCI	IMM	Add content of AC by immediate value of operand	AC ← AC+ IMM + C;	Yes	Short
SUBCI	IMM	Subtract content of AC by immediate value of operand	AC ← AC – IMM – C;	Yes	Short
EQI	IMM	Equal comparison with immediate value of operand	if (AC == IMM) AC ← 1; else AC ← 0;	No	Short
NEQI	IMM	Not equal comparison with immediate value of operand	if (AC != IMM) AC ← 1; else AC ← 0;	No	Short
GTI	IMM	Greater than comparison with immediate value of operand	if (AC > IMM) AC ← 1; else AC ← 0;	No	Short
GEI	IMM	Greater or equal comparison with immediate value of operand	if (AC >= IMM) AC ← 1; else AC ← 0;	No	Short
LTI	IMM	Less than comparison with immediate value of operand	if (AC < IMM) AC ← 1; else AC ← 0;	No	Short
LEI	IMM	Less or equal comparison with immediate value of operand	if (AC <= IMM) AC ← 1; else AC ← 0;	No	Short

ตารางที่ ก.2 อธิบายความหมายและการทำงานของคำสั่งประเภทเลขคณิตและตรรกะ (ต่อ)

Mnemonic	Operand	Description	Operation	C	Format
SHL0	-	Shift content of AC left one bit (0 is shifted into AC)	AC ← {AC[30:0], 1'b0};	Yes	Short
SHR0	-	Shift content of AC right one bit (0 is shifted into AC)	AC ← {1'b0, AC[31:1]};	Yes	Short
SHL	-	Circular shift content of AC left one bit	AC ← {AC[30:0], AC[31]};	No	Short
SHR	-	Circular shift content of AC left one bit	AC ← {AC[0], AC[31:1]};	No	Short
SHLC	-	Shift content of AC left one bit (Carry bit is shifted into AC)	AC ← {AC[30:0], C};	Yes	Short
SHRC	-	Shift content of AC right one bit (Carry bit is shifted into AC)	AC ← {C, AC[31:1]};	Yes	Short
SHL4	-	Circular shift content of AC left four bit	AC ← {AC[27:0], AC[31:28]};	No	Short
SHR4	-	Circular shift content of AC left four bit	AC ← {AC[3:0], AC[31:4]};	No	Short
NOT	-	Bitwise not on content of AC	AC ← ~AC;	No	Short

ตารางที่ ก.3 อธิบายความหมายและการทำงานของคำสั่งประเภทพิเศษ

Mnemonic	Operand	Description	Operation	Format
LIT	lit8bit	Put the literal 8 bit in AC	AC ← lit8bit;	Short
LITL	lit24bit	Put the literal 24 bit in AC	AC ← lit24bit;	Short
NOP	-	No operation	-	Long1
BLANK	-	Reserved opcode for instruction packing's mechanism	-	Short

ตารางที่ ก.4 อธิบายความหมายและการทำงานของคำสั่งประเภทจัดกการข้อมูล

Mnemonic	Operand	Description	Operation	Format
LD	ADS	Load content of memory into AC	AC ← M[ADS];	Short
LDL	ADS	Load content of memory into AC	AC ← M[ADS];	Long1
ST	ADS	Store content of AC into memory	M[ADS] ← AC;	Short
STL	ADS	Store content of AC into memory	M[ADS] ← AC;	Long1
LDX	DISP	Load index element of array indicated by AC	AC ← M[AC+DISP];	Short
STV	LVAR	Store indicated local variable into memory	M[AC] ← M[FP-LVAR];	Short
GET	LVAR	Get content of local variable into AC	AC ← M[FP-LVAR];	Short
PUT	LVAR	Put content of AC into local variable	M[FP-LVAR] ← AC;	Short
PASS	IMM	Pass subroutine's parameter	AC ← M[FP+IMM];	Short
INC	LVAR	Increment local variable by one	M[FP+IMM] ← M[FP+IMM] + 1;	Short
DEC	LVAR	Decrement local variable by one	M[FP+IMM] ← M[FP+IMM] – 1;	Short

ตารางที่ ก.5 อธิบายความหมายและการทำงานของคำสั่งประเภทควบคุม

Mnemonic	Operand	Extra operand	Description	Operation	Format
JMP	OFFSET	-	Unconditional jump	PC ← PC + OFFSET;	Short
JMPL	OFFSET	-	Unconditional jump	PC ← PC + OFFSET;	Long1
JT	OFFSET	-	Jump if true (AC != 0)	if(AC != 0) PC ← PC + OFFSET;	Short
JTL	OFFSET	-	Jump if true (AC != 0)	if(AC != 0) PC ← PC + OFFSET;	Long1
JF	OFFSET	-	Jump if false (AC = 0)	if(AC == 0) PC ← PC + OFFSET;	Short
JFL	OFFSET	-	Jump if false (AC = 0)	if(AC == 0) PC ← PC + OFFSET;	Long1
JC	OFFSET	-	Jump if carry (C = 1)	if(C == 1) PC ← PC + OFFSET;	Short
JCL	OFFSET	-	Jump if carry (C = 1)	if(C == 1) PC ← PC + OFFSET;	Long1
JNC	OFFSET	-	Jump if not carry (C = 0)	if(C == 0) PC ← PC + OFFSET;	Short
JNCL	OFFSET	-	Jump if not carry (C = 0)	if(C == 0) PC ← PC + OFFSET;	Long1
IJX	OFFSET	LVAR	Increment and jump	M[FP-LVAR] ← M[FP-LVAR] + 1; if(AC > (M[FP-LVAR]+1)) PC ← PC + OFFSET;	Long2
CALL	ADS	FS	Call subroutine	PC ← ADS; FP ← FP + FS; M[FP+FS] ← PC;	Long2
RET	FS	-	Return from subroutine	PC ← M[FP]; FP ← FP - FS;	Short

ก.2 การให้รหัสดำเนินการ

ส่วนนี้แสดงการให้รหัสดำเนินการแก่คำสั่งในชุดคำสั่ง แสดงในตารางที่ ก.6

ตารางที่ ก.6 รหัสดำเนินการของคำสั่งทั้งหมดของชุดคำสั่งนี้

	Short format								Long1 format				Long2 format			
	00_XXXXXX				01_XXXXXX				10_XXXXXX				11_XXXXXX			
	0X	1X	2X	3X	4X	5X	6X	7X	8X	9X	AX	BX	CX	DX	EX	FX
0	ADDI		ADDCI		ADD	SHL0	ADDC	SHLC	NOP				CALL		IJX	
1	SUBI		SUBCI		SUB	SHR0	SUBC	SHRC					CALL		IJX	
2	MULI				MUL	SHL							CALL		IJX	
3	DIVI				DIV	SHR							CALL		IJX	
4	ORI				OR	SHL4							CALL		IJX	
5	ANDI				AND	SHR4							CALL		IJX	
6	ADDI				XOR	NOT							CALL		IJX	
7	SUBI				MOD								CALL		IJX	
8	NEQI	INC	LD	RET	NEQ						LDL		CALL		IJX	
9	EQI	DEC	ST	JMP	EQ						STL	JMPL	CALL		IJX	
A	GTI	LIT	LDX	JT	GT					LITL		JTL	CALL		IJX	
B	GEI		STV	JF	GE							JFL	CALL		IJX	
C	LTI		GET	JC	LT							JCL	CALL		IJX	
D	LEI		PUT	JNC	LE							JNCL	CALL		IJX	
E			PASS										CALL		IJX	
F				BLANK									CALL		IJX	

ภาคผนวก ข

สถานะของหน่วยประมวลผล

ในส่วนนี้แสดงรายละเอียดเพิ่มเติมเนื้อหาในบทที่ 3 เกี่ยวกับสถานะการทำงานของวงจรหน่วยประมวลผล

วงจรหน่วยประมวลผลนี้มีสถานะการทำงานทั้งหมด 21 สถานะ ชื่อของสถานะต่างๆนั้นจะได้รับสัญลักษณ์ เพื่อความกะทัดรัดในการกล่าวอ้างถึง ดังแสดงในตารางที่ ข.1

ตารางที่ ข.1 สัญลักษณ์ที่ใช้แทนชื่อสถานะของหน่วยประมวลผล

สัญลักษณ์	ชื่อสถานะ
ST_IFD	สถานะดึงคำสั่งและถอดรหัสคำสั่ง
ST_BOP	สถานะกระทำคำสั่งการดำเนินการคู่แบบตัวแปรเฉพาะที่
ST_BOPI	สถานะกระทำคำสั่งการดำเนินการคู่แบบทันที
ST_LD	สถานะกระทำคำสั่งโหลด
ST_ST	สถานะกระทำคำสั่งจัดเก็บ
ST_LDX1	สถานะกระทำคำสั่งโหลดตัวชี้ที่ 1
ST_LDX2	สถานะกระทำคำสั่งโหลดตัวชี้ที่ 2
ST_STV1	สถานะกระทำคำสั่งจัดเก็บตัวแปรที่ 1
ST_STV2	สถานะกระทำคำสั่งจัดเก็บตัวแปรที่ 2
ST_GET	สถานะกระทำคำสั่งดึง
ST_PUT	สถานะกระทำคำสั่งวาง
ST_INC1	สถานะกระทำคำสั่งเพิ่มลดที่ 1
ST_INC2	สถานะกระทำคำสั่งเพิ่มลดที่ 2
ST_CALL	สถานะกระทำคำสั่งเรียกโปรแกรมย่อย
ST_RET	สถานะกระทำคำสั่งกลับสู่โปรแกรมหลัก
ST_JMP	สถานะกระทำคำสั่งกระโดด
ST_IJX1	สถานะกระทำคำสั่งเพิ่มแล้วกระโดดที่ 1
ST_IJX2	สถานะกระทำคำสั่งเพิ่มแล้วกระโดดที่ 2
ST_PASS	สถานะกระทำคำสั่งส่งพารามิเตอร์
ST_NOP	สถานะไม่มีการดำเนินการ
ST_EXIFD	สถานะดึงคำสั่งและถอดรหัสคำสั่งแบบพิเศษ

พฤติกรรมการทำงานของหน่วยประมวลผล ขึ้นอยู่กับคำสั่งที่กระทำ โดยในการทำงานทุกคำสั่งจะมีการทำงานในขั้นตอนแรกที่เหมือนกันคือ ขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่ง (Instruction fetch & decode phase) เมื่อทำงานในขั้นตอนนี้เสร็จ หน่วยประมวลผลจะทำงานในสถานะสำหรับกระทำคำสั่งที่อ่านมาได้ ดังนั้นสถานะถัดไปของสถานะดึงคำสั่งและถอดรหัสคำสั่ง (ST_IFD) จึงขึ้นอยู่กับคำสั่งที่อ่านได้จากหน่วยความจำ ดังแสดงในตารางที่ ข.2

ตารางที่ ข.2 แสดงสถานะถัดไปของสถานะดึงคำสั่งและถอดรหัสคำสั่ง

คำสั่งที่อ่านได้	สถานะถัดไป
ADD, SUB, MUL, DIV, OR, AND, XOR, MOD, ADDC, SUBC, EQ, NEQ, GT, GE, LT, LE	ST_BOP
ADDI, SUBI, MULI, DIVI, ORI, ANDI, XORI, MODI, ADDCI, SUBCI, EQI, NEQI, GTI, GEI, LTI, LEI	ST_BOPI
SHL0, SHR0, SHL, SHR, SHLC, SHRC, SHL4, SHR4, NOT	ST_BOPI
LD, LDL	ST_LD
ST, STL	ST_ST
LDX	ST_LDX1
STV1	ST_STV1
GET	ST_GET
PUT	ST_PUT
INC, DEC	ST_INC1
CALL	ST_CALL
RET	ST_RET
JMP, JMPL, JT, JTL, JF, JFL, JC, JCL, JNC, JNCL	ST_JMP
IJX	ST_IJX1
PASS	ST_PASS
LIT, LITL	ST_BOPI
NOP	ST_NOP

พฤติกรรมการทำงานของแต่ละสถานะแสดงในตารางที่ ข.3 ในการอธิบายพฤติกรรมการทำงานของสถานะแต่ละสถานะ จะมีการใช้สัญลักษณ์ในตารางที่ ก.1 เพื่อความกะทัดรัดของเนื้อหา

ตารางที่ ข.3 พฤติกรรมการขนถ่ายข้อมูลระหว่างรีจิสเตอร์ในสถานะต่างๆ

สัญลักษณ์	ชื่อสถานะ	สถานะถัดไป
ST_IFD	PC ← PC + 1; IR ← M[PC];	แสดงในตารางที่ ข.2
ST_BOP	AC ← AC op M[FP-LVAR];	ST_IFD
ST_BOPI	AC ← AC op IMM;	ST_IFD
ST_LD	AC ← M[ADS];	ST_IFD
ST_ST	M[ADS] ← AC;	ST_IFD
ST_LDX1	AC ← AC + DISP;	ST_LDX2
ST_LDX2	AC ← M[AC];	ST_IFD
ST_STV1	TEMP ← M[FP-LVAR];	ST_STV2
ST_STV2	M[AC] ← TEMP;	ST_IFD
ST_GET	AC ← M[FP-LVAR];	ST_IFD
ST_PUT	M[FP-LVAR] ← AC;	ST_IFD
ST_INC1	TEMP ← M[FP-LVAR] + 1;	ST_INC2
ST_INC2	M[FP-LVAR] ← TEMP;	ST_IFD
ST_CALL	PC ← ADS; M[FP+FS] ← PC; FP ← FP + FS;	ST_IFD
ST_RET	PC ← M[FP]; FP ← FP – FS;	ST_IFD
ST_JMP	If condition is met then PC ← PC + OFFSET;	ST_IFD
ST_IJX1	TEMP ← M[FP-LVAR] + 1;	ST_IJX2
ST_IJX2	M[FP-LVAR] ← TEMP; if(AC > TEMP) then PC ← PC + OFFSET;	ST_IFD
ST_PASS	M[FP+IMM] ← AC;	ST_IFD
ST_NOP	-	ST_IFD

ประวัติผู้เขียนวิทยานิพนธ์

นายเฉลิมพงศ์ สัตยาวิบูล เกิดเมื่อวันที่ 27 ตุลาคม พ.ศ. 2527 ที่จังหวัดกรุงเทพฯ สำเร็จการศึกษาระดับมัธยมศึกษาจากโรงเรียนหอวัง เข้าศึกษาต่อในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2545 สำเร็จการศึกษาระดับปริญญาบัณฑิต ในสาขาวิชาวิศวกรรมคอมพิวเตอร์ จากคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยในปีการศึกษา 2548