Green Sheet: Pipelain ISA

Carlos Mata ,Felipe Vargas, Ignacio Grané

Instruction Set Summary

Instruction	Description
`sum`	Adds two registers or register-inmediate and stores on an register
`com`	Compare two registes or register-inmediate
`set`	Set the value register or inmediate on an register
`dec`	Decrement two registers or register-inmediate and stores on an register
`divi`	Divition of powers of 2
`xor`	Logical XOR operation with two registers or register-inmediate and stores the result
`aset`	Set the value register or inmediate inverse on an register
`stw`	Stores a word from destination value in memory
`ldw`	Loads a word to destination value from memory
`savepix`	Save the pixel on the memmory
`letter`	Loads the byte of the char value from text
`end`	Finshed the program
`nop`	Make an nop

Instruction Formats

Format	Field	Description
Data Processing Register (DPR)	`cond`	Condition code
	`opcode (op)`	Operation code
	`s`	Set condition codes
	`operand1`	First operand register
	`destination`	Destination register
	`operand2`	Second operand register
Data Processing Immediate (DPI)	`cond`	Condition code
	`opcode (op)`	Operation code
	`s`	Set condition codes
	`operand1`	First operand register
	`destination`	Destination register
	`imm8`	Immediate value
Memory Register (MR)	`cond`	Condition code
	`opcode (op)`	Operation code
	`b`	Byte/Wodestination bit
	`w`	Write-back bit
	`l`	Load/Store bit
	`operand1`	Base register
	`destination`	Destination register
	`operand2`	Second operand register
Memory Immediate Offset (MIO)	`cond`	Condition code
	`opcode (op)`	Operation code
	`b`	Byte/Wodestination bit
	`l`	Load/Store bit
	`operand1`	Base register
	`destination`	Destination register
	`imm12`	Immediate offset
Branch (B)	`cond`	Condition code
	`l`	Link bit
	`imm24`	Signed offset (24 bits)
Special (S)	`cond`	Condition code
	`opcode (op)`	Operation code
	`funct`	Function to execute

Conditional Flags

cond	Mnemonic	Description
0000	eq	Equal
0001	ne	Not equal
1010	ge	Signed greater than or equal
1011	lt	Signed less than
1101	le	Signed less than or equal
1110		Always
1111		Special Instruction

Registers

Binary	Name	Use
0000	r0	Argument / retuoperand1 value / temporary variable
0001	r1	Argument / temporary variable
0010	r2	Argument / temporary variable
0011	r3	Argument / temporary variable
0100	r4	Saved variable
0101	r5	Saved variable
0110	r6	Saved variable
0111	r7	Saved variable
1000	r8	Saved variable
1001	r9	Saved variable
1010	r10	Saved variable
1011	r11	Saved variable
1100	r12	Temporary variable
1101	r13 (sp)	Stack Pointer
1110	r14 (lr)	Link Register
1111	r15 (pc)	Program Counter

DATA-PROCESSING INSTRUCTIONS

funct	Example	Description	Operation
0100	sum, destination, operand1, operand2	Add	destination = operand1 + operand2
1010 (s = 1)	com, operand1, operand2	Compare	Set flags based on operand1 - operand2
1101	set, destination, operand2	Move	destination = operand2
0010	dec, destination, operand1, operand2	Subtract	destination = operand1 - operand2
1101 (i = 0 and sb = 10)	divi, destination, operand2, rs/shamt5	Arithmetic Shift Right	destination = operand2≫operand2
0001	eor destination, operand1, operand2	Bitwise XOR	destination = operand1 ⊕ operand2

Core Instructions

CORE INSTRUCTION FORMAT DATA

31:28	27:26	25:20	19:16	15:12	11:8	7	6	5	4	3	2	0
cond	op 00	funct	operand1	destination	rot 11:8	imm8 7:0						
cond	op 00	funct	operand1	destination	shamt5 11:7		sh		l	operand2		

CORE INSTRUCTION FORMAT MEMORY

31	28	27	26	25	20	19	16	15	12	11	7	6	5	4	3	2	0
cond		op 01		funct		operand1		destination		imm12 11:0							
cond		op 01		funct		operand1		destination		shamt5 11:7		sh		1	operand2		

CORE INSTRUCTION FORMAT BRANCH

31	30	29	28	27	26	25	24	23	0
cond				op		func		imm24	

CORE INSTRUCTION FORMAT SPECIAL

31:4	3:0
1111 1111 1111 11111 11111 11111	func

MEMORY INSTRUCTIONS

op	b	l	Example	Description	Operation
01	0	0	stw, destination, operand1, operand2	Store by word	mem[*adr*] ← destination
01	0	1	ldw, destination, operand1, operand2	Load by word	destination ← mem[*adr*]
01	1	0	savepix, destination, operand1, operand2	Save pixel value	mem[*adr*] ← destination$_{7:0}$
01	1	1	letter, destination, operand1, operand2	Get the letter	destination ← mem[*adr*]$_{7:0}$

BRANCH INSTRUCTIONS

l	Use	Description	Operation
0	b, label	Branch	pc ← (pc+8)+imm24≪2
1	bl, label	Branch with Link	lr ← (pc+8)-4; pc ← (pc+8)+imm24≪2

SPECIAL INSTRUCTIONS

funct	Example	Description
1111	end,	End the program
1110	nop,	Make an nop