ELEC6027 - VLSI Design Project : Programmers Guide

Team R4

14th April, 2014

1 Introduction

Lorem Ipsum. . .

2 Architecture

Lorem Ipsum. . .

3 Register Description

Lorem Ipsum. . .

4 Instruction Set

The complete instruction set architecture includes a number of instructions for performing calculations on data, moving data between external memory and general purpose registers, transfer of control within a program and interrupt handling. It is based around a RISC architecture and as such has a highly orthogonal formatting of bit fields within the instruction code.

All instruction implemented by the architecture fall into one of 6 groups:

- Data Manipulation
- Byte Immediate
- Data Transfer
- Control Transfer
- Stack Operations
- Interrupts

Each instruction has only one addressing mode associated with it, determined by which group it falls within. Data manipulation instructions have either a register-register or register-immediate addressing mode for performing arithmetic, logic or shift operations. Byte immediate instructions have a register-immediate addressing mode for arithmetic and load immediate type operations. Data transfer instructions have a base plus offset addressing mode for accessing external memory using an address stored in a GPR. Control transfer instructions have PC relative, register indirect and base plus offset addressing modes for changing the value of the program counter. Stack operations have register indirect preincrement or register indirect postdecrement addressing modes for accessing external memory and adjusting the stack pointer value. While interrupt operations have register indirect with postdecrement or preincrement addressing modes for restoring program counter and accessing the stack.

4.1 General Instruction Formatting

	Instruction Type	**Sub-Type**	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A1	**Data Manipulation**	**Register**	Opcode					Rd			Ra			Rb			X	X
A2		**Immediate**						Rd			Ra			imm4/5				
B	**Byte Immediate**		Opcode					Rd			imm8							
C	**Data Transfer**		0	LS	0	0	0	Rd			Ra			imm5				
D1	**Control Transfer**	**Others**	1	1	1	1	0	Cond.			imm8							
D2		**Jump**									Ra			imm5				
E	**Stack Operations**		0	U	0	0	1	L	X	X	Ra			0	0	0	0	1
F	**Interrupts**		1	1	0	0	1	ICond.			1	1	1	X	X	X	X	X

Instruction Field Definitions

Opcode:	Operation code as defined for each instruction
Rd:	Destination Register
Ra:	Source register 1
Rb:	Source register 2
immX:	Immediate value of length X
Cond.:	Branching condition code as defined for branch instructions
ICond.:	Interrupt instruction code as defined for interrupt instructions
LS:	0=Load Data, 1=Store Data
U:	1=PUSH, 0=POP
L:	1=Use Link Register, 0=Use GPR

Pseudocode Notation

Symbol	**Meaning**
$\leftarrow$, $\rightarrow$	Assignment
Result[x]	Bit x of result
Ra[$x : y$]	Bit range from x to y of register Ra
$+Ra$	Positive value in Register Ra
$-Ra$	Negative value in Register Ra
$<$	Numerically greater than
$>$	Numerically less than
$<<$	Logical shift left
$>>$	Logical shift right
$>>>$	arithmetic shift right
Mem[val]	Data at memory location with address val
$\{x,\ y\}$	Contatenation of x and y to form a 16-bit value
($cond$)?	Operation performed if $cond$ evaluates to true
!	Bitwise Negation

Use of the word UNPREDICTABLE indicates that the resultant flag value after operation execution will not be indicative of the ALU result. Instead its value will correspond to the result of an undefined arithmetic operation and as such should not be used.

4.2 ADD Add Word

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	Rd			Ra			Rb			X	X

Syntax

ADD Rd, Ra, Rb eg. ADD R5, R3, R2

Operation

Rd $\leftarrow$ Ra + Rb

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Ra, +Rb, -Result) or
(-Ra, -Rb, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or
(Result $< -2^{16}$) then 1, else 0

Description

The 16-bit word in GPR[Ra] is added to the 16-bit word in GPR[Rb] and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.3 ADDI Add Immediate

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	Rd			Ra			imm5				

Syntax

ADDI Rd, Ra, #imm5 eg. ADDI R5, R3, #7

Operation

Rd $\leftarrow$ Ra + #imm5

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Ra, +#imm5, -Result) or

(-Ra, -#imm5, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

The 16-bit word in GPR[Ra] is added to the sign-extended 5-bit value given in the instruction and the result is placed into GPR[Rd].

Addressing Mode: Register Immediate.

4.4 ADDIB — Add Immediate Byte

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	Rd			imm8							

Syntax

ADDIB Rd, #imm8 eg. ADDIB R5, #93

Operation

$\text{Rd} \leftarrow \text{Rd} + \#\text{imm8}$

$\text{N} \leftarrow$ if Result < 0 then 1, else 0

$\text{Z} \leftarrow$ if Result $= 0$ then 1, else 0

$\text{V} \leftarrow$ if (+Rd, +#imm8, -Result) or
(-Rd, -#imm8, +Result) then 1, else 0

$\text{C} \leftarrow$ if (Result $> 2^{16} - 1$) or
(Result $< -2^{16}$) then 1, else 0

Description

The 16-bit word in GPR[Rd] is added to the sign-extended 8-bit value given in the instruction and the result is placed into GPR[Rd].

Addressing Mode: Register Immediate.

4.5 ADC — Add Word With Carry

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Rd			Ra			Rb			X	X

Syntax

ADC Rd, Ra, Rb eg. ADC R5, R3, R2

Operation

Rd $\leftarrow$ Ra + Rb + C

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Ra, +(Rb+CFlag), -Result) or
(-Ra, -(Rb+CFlag), +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or
(Result $< -2^{16}$) then 1, else 0

Description

The 16-bit word in GPR[Ra] is added to the 16-bit word in GPR[Rb] with the added carry in set according to the Carry flag from previous operation, and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.6 ADCI Add Immediate With Carry

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	Rd			Ra			imm5				

Syntax

ADCI Rd, Ra, #imm5 eg. ADCI R5, R4, #7

Operation

$\text{Rd} \leftarrow \text{Ra} + \#\text{imm5} + \text{C}$

$\text{N} \leftarrow \text{if Result} < 0 \text{ then } 1, \text{ else } 0$

$\text{Z} \leftarrow \text{if Result} = 0 \text{ then } 1, \text{ else } 0$

$\text{V} \leftarrow$ if (+Ra, +(#imm5+CFlag), -Result) or

(-Ra, -(#imm5+CFlag), +Result) then 1, else 0

$\text{C} \leftarrow \text{if } (\text{Result} > 2^{16} - 1) \text{ or}$

$(\text{Result} < -2^{16}) \text{ then } 1, \text{ else } 0$

Description

The 16-bit word in GPR[Ra] is added to the sign-extended 5-bit value given in the instruction with carry in set according to the Carry flag from previous operation, and the result is placed into GPR[Rd].

Addressing Mode: Register Immediate.

4.7 NEG — Negate Word

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	Rd			Ra			Rb			X	X

Syntax

NEG Rd, Ra — eg. NEG R5, R3

Operation

Rd $\leftarrow$ 0 - Ra

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

The 16-bit word in GPR[Ra] is added to the 16-bit word in GPR[Rb] and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.8 SUB — Subtract Word

Format

15	14	13	12	11	10 9 8	7 6 5	4 3 2	1	0
0	1	0	1	0	Rd	Ra	Rb	X	X

Syntax

SUB Rd, Ra, Rb — eg. SUB R5, R3, R2

Operation

Rd ← Ra - Rb

N ← if Result < 0 then 1, else 0

Z ← if Result $= 0$ then 1, else 0

V ← if (+Ra, +Rb, -Result) or
(-Ra, -Rb, +Result) then 1, else 0

C ← if (Result $> 2^{16} - 1$) or
(Result $< -2^{16}$) then 1, else 0

Description

The 16-bit word in GPR[Rb] is subtracted from the 16-bit word in GPR[Ra] and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.9 SUBI — Subtract Immediate

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	Rd			Ra			imm5				

Syntax

SUBI Rd, Ra, #imm5 — eg. SUBI R5, R3, #7

Operation

Rd $\leftarrow$ Ra - #imm5

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Ra, +#imm5, -Result) or

(-Ra, -#imm5, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

The sign extended 5-bit value given in the instruction is subtracted from the 16-bit word in GPR[Ra] and the result is placed into GPR[Rd].

Addressing Mode: Register Immediate.

4.10 SUBIB Subtract Immediate Byte

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	Rd			imm8							

Syntax

SUBIB Rd, #imm8 eg. SUBIB R5, #93

Operation

Rd $\leftarrow$ Rd - #imm8

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Rd, +#imm8, -Result) or

(-Rd, -#imm8, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

The 8-bit immediate value given in the instruction is subtracted from the 16-bit word in GPR[Rd] and the result is placed into GPR[Rd].

Addressing Mode: Register Immediate.

4.11 SUC Subtract Word With Carry

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	Rd			Ra			Rb			X	X

Syntax

SUC Rd, Ra, Rb eg. SUC R5, R3, R2

Operation

Rd $\leftarrow$ Ra - Rb - C

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Ra, +(Rb-CFlag), -Result) or

(-Ra, -(Rb-CFlag), +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

The 16-bit word in GPR[Rb] is subtracted from the 16-bit word in GPR[Rb] with the subtracted carry in set according to the Carry flag from previous operation, and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.12 SUCI Subtract Immediate With Carry

Format

15	14	13	12	11	10 9 8	7 6 5	4 3 2 1 0
0	1	1	0	1	Rd	Ra	imm5

Syntax

SUCI Rd, Ra, #imm5 eg. SUCI R5, R4, #7

Operation

Rd $\leftarrow$ Ra - #imm5 - C

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Ra, +(#imm5-CFlag), -Result) or

(-Ra, -(#imm5-CFlag), +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

The 5-bit immediate value in instruction is subtracted from the 16-bit word in GPR[Ra] with the subtracted carry in set according to the Carry flag from previous operation, and the result is placed into GPR[Rd].

Addressing Mode: Register Immediate.

4.13 CMP — Compare Word

Format

15	14	13	12	11	10 9 8	7 6 5	4 3 2	1	0
0	0	1	1	1	Rd	Ra	Rb	X	X

Syntax

CMP Ra, Rb — eg. CMP R3, R2

Operation

Ra - Rb

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ if (+Ra, +Rb, -Result) or
(-Ra, -Rb, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or
(Result $< -2^{16}$) then 1, else 0

Description

The 16-bit word in GPR[Rb] is subtracted from the 16-bit word in GPR[Ra] and the status flags are updated without saving the result.

Addressing Mode: Register Register.

4.14 CMPI — Compare Immediate

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	Rd			Ra			imm5				

Syntax

CMPI Ra, #imm5 eg. CMPI R3, #7

Operation

Ra - #imm5

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Ra, +#imm5, -Result) or
(-Ra, -#imm5, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or
(Result $< -2^{16}$) then 1, else 0

Description

The sign extended 5-bit value given in the instruction is subtracted from the 16-bit word in GPR[Ra] and the status flags are updated without saving the result.

Addressing Mode: Register Immediate.

4.15 AND Logical AND

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	Rd			Ra			Rb			X	X

Syntax

AND Rd, Ra, Rb eg. AND R5, R3, R2

Operation

Rd ← Ra AND Rb

N ← if Result $<$ 0 then 1, else 0

Z ← if Result $=$ 0 then 1, else 0

V ← UNPREDICTABLE

C ← UNPREDICTABLE

Description

The logical AND of the 16-bit words in GPR[Ra] and GPR[Rb] is performed and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.16 OR — Logical OR

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	Rd			Ra			Rb			X	X

Syntax

OR Rd, Ra, Rb — eg. OR R5, R3, R2

Operation

Rd ← Ra OR Rb

N ← if Result < 0 then 1, else 0

Z ← if Result $= 0$ then 1, else 0

V ← UNPREDICTABLE

C ← UNPREDICTABLE

Description

The logical OR of the 16-bit words in GPR[Ra] and GPR[Rb] is performed and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.17 XOR — Logical XOR

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	Rd			Ra			Rb			X	X

Syntax

XOR Rd, Ra, Rb — eg. XOR R5, R3, R2

Operation

Rd $\leftarrow$ Ra XOR Rb

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ UNPREDICTABLE

C $\leftarrow$ UNPREDICTABLE

Description

The logical XOR of the 16-bit words in GPR[Ra] and GPR[Rb] is performed and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.18 NOT Logical NOT

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	Rd			Ra			Rb			X	X

Syntax

NOT Rd, Ra eg. NOT R5, R3

Operation

Rd $\leftarrow$ NOT Ra

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ UNPREDICTABLE

C $\leftarrow$ UNPREDICTABLE

Description

The logical NOT of the 16-bit word in GPR[Ra] is performed and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.19 NAND — Logical NAND

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0		Rd			Ra			Rb		X	X

Syntax

NAND Rd, Ra, Rb — eg. NAND R5, R3, R2

Operation

Rd ← Ra NAND Rb

N ← if Result $<$ 0 then 1, else 0

Z ← if Result $=$ 0 then 1, else 0

V ← UNPREDICTABLE

C ← UNPREDICTABLE

Description

The logical NAND of the 16-bit words in GPR[Ra] and GPR[Rb] is performed and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.20 NOR — Logical NOR

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	Rd			Ra			Rb			X	X

Syntax

NOR Rd, Ra, Rb eg. NOR R5, R3, R2

Operation

Rd $\leftarrow$ Ra NOR Rb

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ UNPREDICTABLE

C $\leftarrow$ UNPREDICTABLE

Description

The logical NOR of the 16-bit words in GPR[Ra] and GPR[Rb] is performed and the result is placed into GPR[Rd].

Addressing Mode: Register Register.

4.21 LSL Logical Shift Left

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	Rd			Ra			0	imm4			

Syntax

LSL Rd, Ra, #imm4 eg. LSL R5, R3, #7

Operation

$\text{Rd} \leftarrow \text{Ra} << \#\text{imm4}$

$\text{N} \leftarrow \text{if Result} < 0 \text{ then } 1, \text{ else } 0$

$\text{Z} \leftarrow \text{if Result} = 0 \text{ then } 1, \text{ else } 0$

$\text{V} \leftarrow \text{UNPREDICTABLE}$

$\text{C} \leftarrow \text{UNPREDICTABLE}$

Description

The 16-bit word in GPR[Ra] is shifted left by the 4-bit amount specified in the instruction, shifting in zeros, and the result is placed into GPR[Rd].

Addressing Mode: Register Immediate.

4.22 LSR Logical Shift Right

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	Rd			Ra			0	imm4			

Syntax

LSR Rd, Ra, #imm4 eg. LSR R5, R3, #7

Operation

Rd $\leftarrow$ Ra >> #imm4

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ UNPREDICTABLE

C $\leftarrow$ UNPREDICTABLE

Description

The 16-bit word in GPR[Ra] is shifted right by the 4-bit amount specified in the instruction, shifting in zeros, and the result is placed into GPR[Rd].

Addressing Mode: Register Immediate.

4.23 ASR Arithmetic Shift Right

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	Rd			Ra			0	imm4			

Syntax

ASR Rd, Ra, #imm4 eg. ASR R5, R3, #7

Operation

Rd $\leftarrow$ Ra >>> #imm4

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ UNPREDICTABLE

C $\leftarrow$ UNPREDICTABLE

Description

The 16-bit word in GPR[Ra] is shifted right by the 4-bit amount specified in the instruction, shifting in the sign bit of Ra, and the result is placed into GPR[Rd].

Addressing Mode: Register Immediate.

4.24 LDW Load Word

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0		Rd			Ra				imm5		

Syntax

LDW Rd, [Ra, #imm5] eg. LDW R5, [R3, #7]

Operation

Rd $\leftarrow$ Mem[Ra + #imm5]

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Ra, +#imm5, -Result) or

(-Ra, -#imm5, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

Data is loaded from memory at the resultant address from addition of GPR[Ra] and the 5-bit immediate value specified in the instruction, and the result is placed into GPR[Rd].

Addressing Mode: Base Plus Offset.

4.25 STW — Store Word

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	Rd			Ra			imm5				

Syntax

STW Rd, [Ra, #imm5] eg. STW R5, [R3, #7]

Operation

$\text{Rd} \rightarrow \text{Mem[Ra + \#imm5]}$

$\text{N} \leftarrow \text{if Result} < 0 \text{ then 1, else 0}$

$\text{Z} \leftarrow \text{if Result} = 0 \text{ then 1, else 0}$

$\text{V} \leftarrow \text{if (+Ra, +\#imm5, -Result) or}$

$\quad\quad \text{(-Ra, -\#imm5, +Result) then 1, else 0}$

$\text{C} \leftarrow \text{if (Result} > 2^{16} - 1\text{) or}$

$\quad\quad \text{(Result} < -2^{16}\text{) then 1, else 0}$

Description

Data in GPR[Rd] is stored to memory at the resultant address from addition of GPR[Ra] and the 5-bit immediate value specified in the instruction.

Addressing Mode: Base Plus Offset.

4.26 LUI — Load Upper Immediate

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	Rd			imm8							

Syntax

LUI Rd #imm8 eg. LUI R5, #93

Operation

Rd $\leftarrow$ {#imm8, 0}

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ UNPREDICTABLE

C $\leftarrow$ UNPREDICTABLE

Description

The 8-bit immediate value provided in the instruction is loaded into the top half in GPR[Rd], setting the bottom half to zero.

Addressing Mode: Register Immediate.

4.27 LLI — Load Lower Immediate

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	Rd			imm8							

Syntax

LLI Rd #imm8 eg. LLI R5, #93

Operation

Rd $\leftarrow$ {Rd[15:8], #imm8}

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ UNPREDICTABLE

C $\leftarrow$ UNPREDICTABLE

Description

The 8-bit immediate value provided in the instruction is loaded into the bottom half in GPR[Rd], leaving the top half unchanged.

Addressing Mode: Register Immediate.

4.28 BR — Branch Always

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	imm8							

Syntax

BR LABEL — eg. BR .loop

Operation

$PC \leftarrow PC + \#imm8$

$N \leftarrow$ if Result < 0 then 1, else 0

$Z \leftarrow$ if Result $= 0$ then 1, else 0

$V \leftarrow$ if (+Rd, +#imm8, -Result) or
(-Rd, -#imm8, +Result) then 1, else 0

$C \leftarrow$ if (Result $> 2^{16} - 1$) or
(Result $< -2^{16}$) then 1, else 0

Description

Unconditionally branch to the resultant address from addition of PC and the 8-bit immediate value specified in the instruction. LABEL can be both a symbolic name or a numeric value, and is capable of jumping forwards or backwards.

Addressing Mode: PC Relative.

4.29 BNE — Branch If Not Equal

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	0	imm8							

Syntax

BNE LABEL — eg. BNE .loop

Operation

PC $\leftarrow$ PC + #imm8 (z==0)?

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ if (+Rd, +#imm8, -Result) or

(-Rd, -#imm8, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

Conditionally branch to the resultant address from addition of PC and the 8-bit immediate value specified in the instruction if zero status flag (Z) equals zero. LABEL can be both a symbolic name or a numeric value, and is capable of jumping forwards or backwards.

Addressing Mode: PC Relative.

4.30 BE — Branch If Equal

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	1	imm8							

Syntax

BE LABEL — eg. BE .loop

Operation

PC $\leftarrow$ PC + #imm8 (z==1)?

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result = 0 then 1, else 0

V $\leftarrow$ if (+Rd, +#imm8, -Result) or

(-Rd, -#imm8, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

Conditionally branch to the resultant address from addition of PC and the 8-bit immediate value specified in the instruction if zero status flag (Z) equals one. LABEL can be both a symbolic name or a numeric value, and is capable of jumping forwards or backwards.

Addressing Mode: PC Relative.

4.31 BLT — Branch If Less Than

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	0	imm8							

Syntax

BLT LABEL — eg. BLT .loop

Operation

PC $\leftarrow$ PC + #imm8 (n&!v OR !n&v)?

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ if (+Rd, +#imm8, -Result) or

(-Rd, -#imm8, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

Conditionally branch to the resultant address from addition of PC and the 8-bit immediate value specified in the instruction if negative status flag and overflow status flag are not equivalent. LABEL can be both a symbolic name or a numeric value, and is capable of jumping forwards or backwards.

Addressing Mode: PC Relative.

4.32 BGE Branch If Greater Than Or Equal

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	imm8							

Syntax

BGE LABEL eg. BGE .loop

Operation

PC $\leftarrow$ PC + #imm8 (n&v OR !n&!v)?

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ if (+Rd, +#imm8, -Result) or

(-Rd, -#imm8, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

Conditionally branch to the resultant address from addition of PC and the 8-bit immediate value specified in the instruction if negative status flag and overflow status flag are equivalent. LABEL can be both a symbolic name or a numeric value, and is capable of jumping forwards or backwards.

Addressing Mode: PC Relative.

4.33 BWL — Branch With Link

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	imm8							

Syntax

BWL LABEL — eg. BWL .loop

Operation

LR $\leftarrow$ PC + 1; PC $\leftarrow$ PC + #imm8

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ if (+Rd, +#imm8, -Result) or
(-Rd, -#imm8, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or
(Result $< -2^{16}$) then 1, else 0

Description

Save the current program counter (PC) value plus one to the link register. Then unconditionally branch to the resultant address from addition of PC and the 8-bit immediate value specified in the instruction. LABEL can be both a symbolic name or a numeric value, and is capable of jumping forwards or backwards.

Addressing Mode: PC Relative.

4.34 RET **Return**

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	0	imm8							

Syntax

RET eg. RET

Operation

PC $\leftarrow$ LR

N $\leftarrow$ UNPREDICTABLE

Z $\leftarrow$ UNPREDICTABLE

V $\leftarrow$ UNPREDICTABLE

C $\leftarrow$ UNPREDICTABLE

Description

Unconditionally branch to the address stored in the link register (LR).

Addressing Mode: Register Indirect.

4.35 JMP Jump

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	1	imm8							

Syntax

JMP Ra, #imm5 eg. JMP R3, #7

Operation

PC $\leftarrow$ Ra + #imm5

N $\leftarrow$ if Result $<$ 0 then 1, else 0

Z $\leftarrow$ if Result $=$ 0 then 1, else 0

V $\leftarrow$ if (+Rd, +#imm8, -Result) or

(-Rd, -#imm8, +Result) then 1, else 0

C $\leftarrow$ if (Result $> 2^{16} - 1$) or

(Result $< -2^{16}$) then 1, else 0

Description

Unconditionally jump to the resultant address from the addition of GPR[Ra] and the 5-bit immediate value specified in the instruction.

Addressing Mode: Base Plus Offset.

4.36 PUSH Push From Stack

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	L	X	X		Ra		0	0	0	0	1

Syntax

PUSH Ra	eg. PUSH R3
PUSH RL	eg. PUSH RL

Operation

reg $\rightarrow$ Mem[R7]; R7 $\leftarrow$ R7 - 1

N $\leftarrow$ if Result < 0 then 1, else 0

Z $\leftarrow$ if Result $= 0$ then 1, else 0

V $\leftarrow$ UNPREDICTABLE

C $\leftarrow$ UNPREDICTABLE

Description

'reg' corresponds to either a GPR or the link register, the contents of which are stored to the stack using the address stored in the stack pointer (R7). Then Decrement the stack pointer by one.

Addressing Mode: Register Indirect Postdecrement.

4.37 POP Pop From Stack

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	L	X	X	Ra			0	0	0	0	1

Syntax

POP Ra	eg. POP R3
POP RL	eg. POP RL

Operation

$R7 \leftarrow R7 + 1$; $\mathrm{Mem}[R7] \leftarrow \mathrm{reg}$;

$N \leftarrow$ if Result < 0 then 1, else 0

$Z \leftarrow$ if Result $= 0$ then 1, else 0

$V \leftarrow$ UNPREDICTABLE

$C \leftarrow$ UNPREDICTABLE

Description

Increment the stack pointer by one. Then ‘reg’ corresponds to either a GPR or the link register, the contents of which are retrieved from the stack using the address stored in the stack pointer (R7).

Addressing Modes: Register Indirect Preincrement.

4.38 RETI — Return From Interrupt

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	0	1	1	1	X	X	X	X	X

Syntax

RETI eg. RETI

Operation

$PC \leftarrow Mem[R7]$

$N \leftarrow$ UNPREDICTABLE

$Z \leftarrow$ UNPREDICTABLE

$V \leftarrow$ UNPREDICTABLE

$C \leftarrow$ UNPREDICTABLE

Description

Restore program counter to its value before interrupt occured, which is stored on the stack, pointed to be the stack pointer (R7). This must be the last instruction in an interrupt service routine.

Addressing Mode: Register Indirect.

4.39 ENAI Enable Interrupts

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	1	1	1	1	X	X	X	X	X

Syntax

ENAI eg. ENAI

Operation

IntEn Flag $\leftarrow$ 1

$N \leftarrow N$

$Z \leftarrow Z$

$V \leftarrow V$

$C \leftarrow C$

Description

Turn on interrupts by setting interrupt enable flag to true (1).

4.40 DISI — Disable Interrupts

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	0	1	1	1	X	X	X	X	X

Syntax

DISI — eg. DISI

Operation

IntEn Flag $\leftarrow 0$

$N \leftarrow N$

$Z \leftarrow Z$

$V \leftarrow V$

$C \leftarrow C$

Description

Turn off interrupts by setting interrupt enable flag to false (0).

4.41 STF — Store Status Flags

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1	1	1	1	X	X	X	X	X

Syntax

STF — eg. STF

Operation

Mem [R7] $\leftarrow$ {12-bit 0, Z, C, V, N}; R7 $\leftarrow$ R7 - 1;

N $\leftarrow$ N

Z $\leftarrow$ Z

V $\leftarrow$ V

C $\leftarrow$ C

Description

Store contents of status flags to stack using address held in stack pointer (R7). Then decrement the stack pointer (R7) by one.

The addressing more of this instruction is Register Indirect Postdecrement.

4.42 LDF — Load Status Flags

Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	1	0	0	1	1	1	X	X	X	X	X

Syntax

LDF — eg. LDF

Operation

$R7 \leftarrow R7 + 1; \{Z, C, V, N\} \leftarrow Mem[R7][3:0]$

$N \leftarrow N$

$Z \leftarrow Z$

$V \leftarrow V$

$C \leftarrow C$

Description

Increment the stack pointer (R7) by one. Then load content of status flags with lower 4 bits of value retrieved from stack using address held in stack pointer (R7).

The addressing more of this instruction is Register Indirect Preincrement.

5 Programming Tips

Lorem Ipsum. . .

6 Assembler

The current instruction set architecture includes an assembler for converting symbolic sequences into machine code. This chapter outlines the required formatting and available features of this assembler.

6.1 Instruction Formatting

Each instruction must be formatted using the following syntax, here "[. . .]" indicates an optional field:

```
[.LABELNAME] MNEMONIC, OPERANDS, ..., :[COMMENTS]
```

eg. `.loop ADDI, R5, R3, #5 :Add 5 to R3`

Comments may be added by preceding them with either : or ;.

Accepted general purpose register values are: R0, R1, R2, R3, R4, R5, R6, R7, SP. These can be upper or lower case and SP is equivalently evaluated to R7.

Branch instructions can take either a symbolic or numeric value. Where a numeric must be relative and between -32 and 31 for a JMP instruction, or between -128 and 127 for any other branch type. If the branch exceeds the accepted range, the assembler will flag an error message.

All label names must begin with a '.' while .ISR/.isr and .define are special cases used for the interrupt service routine and variable definitions respectively.

Instruction-less or comments only lines are allowed within the assembly file.

Special Case Label
The .ISR/.isr label is reserved for the interrupt service routine and may be located anywhere within the file but must finish with a 'RETI' instruction and be no longer than 126 lines of code. Branches may occur within the ISR, but are not allowed into this subroutine with the exception of a return from a separate subroutine.

6.2 Assembler Directives

Symbolic label names are supported for branch-type instructions. Following the previous syntax definition for '.LABELNAME', they can be used instead of numeric branching provided they branch no further than the maximum distance allowed for the instruction used. Definitions are supported by the assembler. They are used to assign meaningful names to the GPRs to aid with programming. Definitions can occur at any point within the file and create a mapping from that point onwards. Different names can be assigned to the same register, but only one is valid at a time.

The accepted syntax for definitions is:

```
.define NAME REGISTER
```

6.3 Running The Assembler

The assembler reads a '.asm' file and outputs a '.hex' file in hexadecimal format. It is run by typing "./assemble filename" at the command line when in the directory of both the assembler executable and the program assembly file. "filename" does not have to include the .asm file extension. The outputted file is saved to the same directory as the input file.

HSL: I'm going to add an option parser to make the UI a bit easier. This section is likely to change a fair amount

Typing -h or –help instead of the file name will bring up the help menu with version information and basic formatting support.

6.4 Error Messages

Code	Description
ERROR1	Instruction mneumonic is not recognized
ERROR2	Register code within instruction is not recognized
ERROR3	Branch condition code is not recognised
ERROR4	Attempting to branch to undefined location
ERROR5	Instruction mneumonic is not recognized
ERROR6	Attempting to shift by more than 16 or perform a negative shift
ERROR7	Magnitude of immediate value for ADDI, ADCI, SUBI, SUCI, LDW or STW is too large
ERROR8	Magnitude of immediate value for CMPI or JMP is too large
ERROR9	Magnitude of immediate value for ADDIB, SUBIB, LUI or LLI is too large
ERROR10	Attempting to jump more than 127 forward or 128 backwards
ERROR11	Duplicate symbolic link names
ERROR12	Illegal branch to ISR
ERROR13	Multiple ISRs in file
ERROR14	Invalid formatting for .define directive

7 Programs

Lorem Ipsum…

8 Simulation

8.1 Running the simulations

Describe sim.py

What it does, why it is needed

How to run for each of the behavioural, extracted and mixed

NEED TO CHANGE SIM.PY TO RUN USING IAINS STRUCTURE (/home/user/design/fcde...)

Clock cycles for each of the programs

Register window - need to do one. Description of also.