ELEC6027 - VLSI Design Project : Programmers Guide

Team R4

April 4, 2014

1 Introduction

Lorem Ipsum...

2 Architecture

Lorem Ipsum...

3 Register Description

Lorem Ipsum...

4 Instruction Set

This chapter describes the full set of instructions implemented by the architecture. Each instruction is divided into one of 6 groups:

- Data Manipulation
- Byte Immediate
- Data Transfer
- Control Transfer

- Stack Operations
- Interrupt Operations

These follow a general formatting as follows:

	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**	Opcode					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**	1	1	1	1	0				Ra			imm5				

4.1 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 $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ if $(+Ra, +Rb, -Result)$ or
$(-Ra, -Rb, +Result)$ then 1, else 0
C $\leftarrow$ if $(Result > 2^{15})$ or
$(Result < -2^{16})$ then 1, else 0

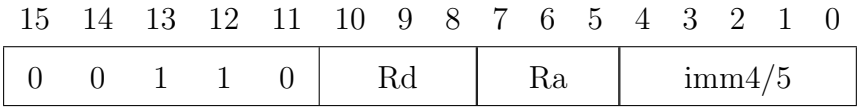
Description

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.

The addressing mode of this instruction is Register Register.

4.2 ADDI — Add Immediate

Format



Syntax

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

Operation

$Rd \leftarrow Ra + \#imm5$
N $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ if $(+Ra, +\#imm5, -Result)$ or
$(-Ra, -\#imm5, +Result)$ then 1, else 0
C $\leftarrow$ if $(Result > 2^{15})$ or
$(Result < -2^{16})$ then 1, else 0

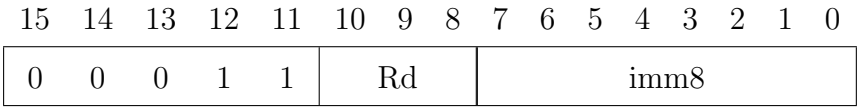
Description

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.

The addressing mode of this instruction is Register Immediate.

4.3 ADDIB — Add Immediate Byte

Format



Syntax

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

Operation

$Rd \leftarrow Rd + \#imm8$
N $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ if $(+Rd, +\#imm8, -Result)$ or
$(-Rd, -\#imm8, +Result)$ then 1, else 0
C $\leftarrow$ if $(Result > 2^{15})$ 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.

The addressing mode of this instruction is Register Immediate.

4.4 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 $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ if $(+Ra, +(Rb + CFlag), -Result)$ or
$(-Ra, -(Rb + CFlag), +Result)$ then 1, else 0
C $\leftarrow$ if $(Result > 2^{15})$ 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.

The addressing mode of this instruction is Register Register.

4.5 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			imm4/5				

Syntax

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

Operation

$Rd \leftarrow Ra + \#imm5 + C$
N $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ if $(+Ra, +(\#imm5 + CFlag), -Result)$ or
$(-Ra, -(\#imm5 + CFlag), +Result)$ then 1, else 0
C $\leftarrow$ if $(Result > 2^{15})$ 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 with carry in set according to the Carry flag from previous operation, and the result is placed into GPR Rd.

The addressing mode of this instruction is Register Immediate.

4.6 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 $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ 0
C $\leftarrow$ if $(Result > 2^{15})$ 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.

The addressing mode of this instruction is Register Register.

4.7 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

$Rd \leftarrow Ra - Rb$
N $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ if $(+Ra, +Rb, -Result)$ or
$(-Ra, -Rb, +Result)$ then 1, else 0
C $\leftarrow$ if $(Result > 2^{15})$ or
$(Result < -2^{16})$ then 1, else 0

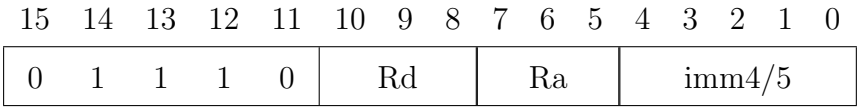
Description

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.

The addressing mode of this instruction is Register Register.

4.8 SUBI Subtract Immediate

Format



Syntax

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

Operation

$Rd \leftarrow Ra - \#imm5$
N $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ if $(+Ra, +\#imm5, -Result)$ or
$(-Ra, -\#imm5, +Result)$ then 1, else 0
C $\leftarrow$ if $(Result > 2^{15})$ or
$(Result < -2^{16})$ then 1, else 0

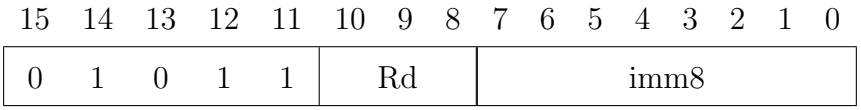
Description

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.

The addressing mode of this instruction is Register Immediate.

4.9 SUBIB Subtract Immediate Byte

Format



Syntax

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

Operation

$Rd \leftarrow Rd - \#imm8$
N $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ if $(+Rd, +\#imm8, -Result)$ or
$(-Rd, -\#imm8, +Result)$ then 1, else 0
C $\leftarrow$ if $(Result > 2^{15})$ 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.

The addressing mode of this instruction is Register Immediate.

4.10 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 $\leftarrow$ Rd[31]

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

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

C $\leftarrow$ if $(Result > 2^{15})$ 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.

The addressing mode of this instruction is Register Register.

4.11 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			imm4/5				

Syntax

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

Operation

$Rd \leftarrow Ra - \#imm5 - C$

N $\leftarrow$ Rd[31]

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

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

C $\leftarrow$ if $(Result > 2^{15})$ 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.

The addressing mode of this instruction is Register Immediate.

4.12 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 $\leftarrow$ Rd[31]

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

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

C $\leftarrow$ if $(Result > 2^{15})$ or
$(Result < -2^{16})$ then 1, else 0

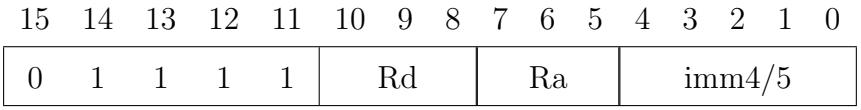
Description

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.

The addressing mode of this instruction is Register Register.

4.13 CMPI Compare Immediate

Format



Syntax

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

Operation

$Ra - \#imm5$
N $\leftarrow$ Rd[31]
Z $\leftarrow$ if $Rd = 0$ then 1, else 0
V $\leftarrow$ if $(+Ra, +\#imm5, -Result)$ or
$(-Ra, -\#imm5, +Result)$ then 1, else 0
C $\leftarrow$ if $(Result > 2^{15})$ or
$(Result < -2^{16})$ then 1, else 0

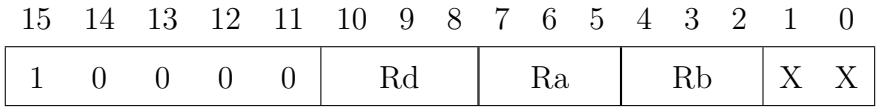
Description

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.

The addressing mode of this instruction is Register Immediate.

4.14 AND Logical AND

Format



Syntax

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

Operation

$Rd \leftarrow Ra\ AND\ Rb$
N ← N
Z ← Z
V ← V
C ← C

Description

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

The addressing mode of this instruction is Register Register.

4.15 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

$Rd \leftarrow Ra\ OR\ Rb$
N ← N
Z ← Z
V ← V
C ← C

Description

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

The addressing mode of this instruction is Register Register.

4.16 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 ← N
Z ← Z
V ← V
C ← C

Description

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

The addressing mode of this instruction is Register Register.

4.17 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 ← N
Z ← Z
V ← V
C ← C

Description

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

The addressing mode of this instruction is Register Register.

4.18 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

$Rd \leftarrow Ra\ NAND\ Rb$
N ← N
Z ← Z
V ← V
C ← C

Description

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

The addressing mode of this instruction is Register Register.

4.19 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 ← N
Z ← Z
V ← V
C ← C

Description

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

The addressing mode of this instruction is Register Register.

4.20 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			imm4/5				

Syntax

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

Operation

$Rd \leftarrow Ra << \#imm4$
N ← N
Z ← Z
V ← V
C ← C

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.

The addressing mode of this instruction is Register Immediate.

4.21 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			imm4/5				

Syntax

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

Operation

$Rd \leftarrow Ra >> \#imm4$
N ← N
Z ← Z
V ← V
C ← C

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.

The addressing mode of this instruction is Register Immediate.

4.22 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			imm4/5				

Syntax

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

Operation

$Rd \leftarrow Ra >>> \#imm4$
N ← N
Z ← Z
V ← V
C ← C

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.

The addressing mode of this instruction is Register Immediate.

4.23 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 ← N
Z ← Z
V ← V
C ← C

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.

The addressing mode of this instruction is Base Plus Offset.

4.24 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

$Mem[Ra + \#imm5] \leftarrow Rd$
N $\leftarrow$ N
Z $\leftarrow$ Z
V $\leftarrow$ V
C $\leftarrow$ C

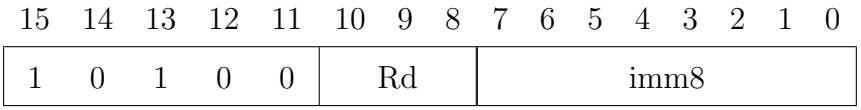
Description

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.

The addressing mode of this instruction is Base Plus Offset.

4.25 LUI Load Upper Immediate

Format



Syntax

LUI Rd #imm8 eg. LUI R5, #93

Operation

$Rd \leftarrow \{\#imm8,\ 0\}$
N ← N
Z ← Z
V ← V
C ← C

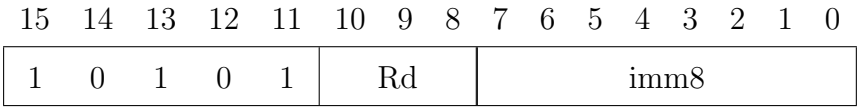
Description

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

The addressing mode of this instruction is Register Immediate.

4.26 LLI Load Lower Immediate

Format



Syntax

LLI Rd #imm8 eg. LLI R5, #93

Operation

$Rd \leftarrow \{Rd[15:8],\ \#imm8\}$
N ← N
Z ← Z
V ← V
C ← C

Description

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

The addressing mode of this instruction is Register Immediate.

4.27 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 ← N
Z ← Z
V ← V
C ← C

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.

The addressing mode of this instruction is PC Relative.

4.28 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 ← N
Z ← Z
V ← V
C ← C

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.

The addressing mode of this instruction is PC Relative.

4.29 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 ← N
Z ← Z
V ← V
C ← C

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.

The addressing mode of this instruction is PC Relative.

4.30 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

$PC \leftarrow PC + \#imm8\ (n\& \sim v\ OR\ \sim n\&n)?$
N ← N
Z ← Z
V ← V
C ← C

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.

The addressing mode of this instruction is PC Relative.

4.31 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

$PC \leftarrow PC + \#imm8\ (n\&v\ OR\ \sim n\& \sim v)?$
N ← N
Z ← Z
V ← V
C ← C

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.

The addressing mode of this instruction is PC Relative.

4.32 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 $\leftarrow$ N

Z $\leftarrow$ Z

V $\leftarrow$ V

C $\leftarrow$ C

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.

The addressing mode of this instruction is PC Relative.

4.33 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 ← N
Z ← Z
V ← V
C ← C

Description

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

The addressing mode of this instruction is Register Indirect.

4.34 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 ← N
Z ← Z
V ← V
C ← C

Description

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

The addressing mode of this instruction is Base Plus Offset.

4.35 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			X	X	X	X	X

Syntax

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

Operation

$Mem[R7] \leftarrow reg$; $R7 \leftarrow R7 - 1$
N $\leftarrow$ N
Z $\leftarrow$ Z
V $\leftarrow$ V
C $\leftarrow$ C

Description

If 'reg' is a general purpose register, store its contents to the stack using the address stored in the stack pointer (R7). If 'reg' is the link register, store its contents to the stack using the address stored in the stack point (R7). Then Decrement the stack pointer by one.

The addressing mode of this instruction is Register Indirect Postdecrement.

4.36 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			X	X	X	X	X

Syntax

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

Operation

$R7 \leftarrow R7 + 1$; $Mem[R7] \leftarrow reg$;
N $\leftarrow$ N
Z $\leftarrow$ Z
V $\leftarrow$ V
C $\leftarrow$ C

Description

Increment the stack pointer by one. Then if 'reg' is a general purpose register, retrieve its contents from the stack using the address stored in the stack pointer (R7). If 'reg' is the link register, retrieve its contents from the stack using the address stored in the stack point (R7).

The addressing modes of this instruction are Register Indirect Preincrement.

4.37 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 $\leftarrow$ N
Z $\leftarrow$ Z
V $\leftarrow$ V
C $\leftarrow$ C

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.

The addressing mode of this instruction is Register Indirect.

4.38 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

$IntEnFlag \leftarrow 1$
N $\leftarrow$ N
Z $\leftarrow$ Z
V $\leftarrow$ V
C $\leftarrow$ C

Description

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

4.39 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

$IntEnFlag \leftarrow 0$
N $\leftarrow$ N
Z $\leftarrow$ Z
V $\leftarrow$ V
C $\leftarrow$ C

Description

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

4.40 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

$Mem[R7] \leftarrow 12 - bit0, Z, C, V, N$
N ← N
Z ← Z
V ← V
C ← C

Description

Store contents of status flags to stack using address held in stack pointer (R7).

CHECK THIS

4.41 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

$Z, C, V, N \leftarrow Mem[R7][3:0]$
N $\leftarrow$ N
Z $\leftarrow$ Z
V $\leftarrow$ V
C $\leftarrow$ C

Description

Load content of status flags with lower 4 bits of value retrieved from stack using address held in stack pointer (R7).

CHECK THIS

5 Assembler

Lorem Ipsum...

6 Programs

Lorem Ipsum...

7 Simulation

Lorem Ipsum...