General Instruction Formatting

	Instruction Type	Sub-Type	15 14 13 12 11	10 9	8 7 6	5 4 3	2 1 0
A	Data Manipulation	Register	Opcode	X X	Rb	Ra	Rd
		Immediate		imm4/5			
B	Byte Immediate		Opcode	imm8			Rd
C	Data Transfer		1 1 1 1 0	X LS	Ro	Ra	Rd
D	Control Transfer	Others	1 1 1 1 1	imm8			Cond.
		Absolute		X X X X X		Ra	
E	Stack Operations		1 1 1 0 1	No.	Rc	Rb	Ra

LS: 1 = Load Data, 0 = Store Data No. = 1 or 2 or 3

Instruction Set Summary

	Mnemonic	Syntax	Semantics	Flags	Encoding	Opcode	Cond.
1	ADD	ADD Rd, Ra, Rb	Rd ← Ra + Rb	n, z	A	00001	-
2	ADDI	ADDI Rd, Ra, #imm5	Rd ← Ra + imm5	n, z	A	00010	-
3	ADDIB	ADDIB Rd, #imm8	Rd ← Rd + imm8	n, z	B	00011	-
4	ADC	ADC Rd, Ra, Rb	Rd ← Ra + Rb	c, n, z	A	00100	-
5	ADCI	ADCI Rd, Ra, #imm5	Rd ← Ra + imm5	c, n, z	A	00101	-
6	NEG	NEG Rd	Rd ← -Rd	n, z	A	00110	-
7	SUB	SUB Rd, Ra, Rb	Rd ← Ra - Rb	n, z	A	00111	-
8	SUBI	SUBI Rd, Ra, #imm5	Rd ← Ra - imm5	n, z	A	01000	-
9	SUBIB	SUBIB Rd, #imm8	Rd ← Rd - imm8	n, z	B	01001	-
10	SUC	SUC Rd, Ra, Rb	Rd ← Ra - Rb	c, n, z	A	01010	-
11	SUCI	SUCI Rd, Ra, #imm5	Rd ← Ra - imm5	c, n, z	A	01011	-
12	SUBN	SUBN Ra, Rb	Ra - Rb	n, z	A	01100	-
13	AND	AND Rd, Ra, Rb	Rd ← Ra AND Rb	-	A	01101	-
14	OR	OR Rd, Ra, Rb	Rd ← Ra OR Rb	-	A	01110	-
15	XOR	XOR Rd, Ra, Rb	Rd ← Ra XOR Rb	-	A	01111	-
16	LSL	LSL Rd, Ra, #imm4	Rd ← Ra << imm4	-	A	10000	-
17	LSR	LSR Rd, Ra, #imm4	Rd ← Ra >> imm4	-	A	10001	-
18	ASR	ASR Rd, Ra, #imm4	Rd ← Ra >> imm4	-	A	10010	-
19	LDW	LDW Rd, [Ra,Ro]	Rd ← Mem[Ra + Ro]	-	C	11110	-
20	SDW	SDW Rd, [Ra,Ro]	Mem[Ra + Ro] ← Rd	-	C	11110	-
21	LUI	LUI Rd, #imm8	Rd[15-8] ← imm8	-	B	10011	-
22	LLI	LLI Rd, #imm8	Rd[7-0] ← imm8	-	B	10100	-
23	BR	BR LABEL	PC ← PC + LABEL	-	D	11111	000
24	BNE	BNE LABEL	(z==0)? PC ← PC + LABEL	-	D	11111	001
25	BE	BE LABEL	(z==1)? PC ← PC + LABEL	-	D	11111	010
26	BLT	BLT LABEL	(n==1)? PC ← PC + LABEL	-	D	11111	011
27	BWL	BWL LABEL	LR ← PC PC ← PC + LABEL	-	D	11111	100
28	JMP	JMP	PC ← LR	-	D	11111	101
29	BAB	BAB Ra	PC ← Ra	-	D	11111	110
30	PUSH{1,2,3}	PUSHx Ra, Rb, Rc	SP ← SP + 1 Update stack	-	E	11101	-
31	POP{1,2,3}	POPx Ra, Rb, Rc	SP ← SP - 1 Update stack	-	E	11101	-

Example Coding

Data Manipulation

These operations are performed by the Arithmetic Logic Unit and examples are shown below.

1	ADD R5, R3, R4	R5 ← R3 + R4
2	ADDI R5, R3, #9	R5 ← R3 + 9
4	ADC R5, R3, R4	R5 ← R3 + R4
5	ADCI R5, R3, #9	R5 ← R3 + 9
6	NEG R5	R5 ← -R5
7	SUB R5, R3, R4	R5 ← R3 - R4
8	SUBI R5, R3, #9	R5 ← R3 - 9
10	SUC R5, R3, R4	R5 ← R3 - R4

11	SUCI R5, R3, #9	R5 ← R3 - 9
12	SUBN R3, R4	R3 - R4
13	AND R5, R3, R4	R5 ← R3 AND R4
14	OR R5, R3, R4	R5 ← R3 OR R4
15	XOR R5, R3, R4	R5 ← R3 XOR R4
16	LSL R5, R3, #3	R5 ← R3 << 3
17	LSR R5, R3, #3	R5 ← R3 >> 3
18	ASR R5, R3, #3	R5 ← R3 >> 3

Byte Immediate

These instructions ADD/SUB an 8-bit immediate value from the given register, replacing the result back in that register. Alternatively, the same formatting is used for loading the upper/lower byte of a register with an 8-bit immediate value.

3	ADDIB R5, #150	R5 ← R5 + 150
9	SUBIB R5, #150	R5 ← R5 - 150
21	LUI R5, #150	R5[15-8] ← 150
22	LLI R5, #150	R5[7-0] ← 150

Data Transfer

When loading data, the value at the memory location held in Ra, adds an offset held in Ro, and replaces the returned value in register Rd. When storing data, the same functionality is used, only with data transferring in opposite direction.

19	LDW R5, [R3, R2]	R5 ← Mem[R3 + R2]
20	SDW R5, [R3, R2]	Mem[R3 + R2] ← R5

Control Transfer

This set of instructions adjust the value of the program counter by a relative amount determined by the location of the given label. Conditions are as follows:

- BR – Branch Always – Unconditionally branch to the stated location
- BNE – Branch if not equal – Conditionally branch if zero status flag (z) equals zero
- BE – Branch if equal – Conditionally branch if zero status flag (z) equals one
- BLT – Branch if less than – Conditionally branch if negative status flag (n) equals one
- BWL – Branch with link – Unconditionally branch to stated location, saving PC to link register (LR)
- JMP – Jump – Unconditionally jump to the value stored in the link register (LR)
- BAB – Absolute Branch – Unconditionally branch to the location held in register Ra

Stack Operations

These operations are for popping or pushing up to three register values from the system stack, useful for context saving when an interrupt occurs. PUSH increments stack pointer (SP) and POP decrements stack pointer (SP) 1, 2 or 3 times for a top-down growing stack.