

Instruction Set Summary

	Mnemonic	Syntax	Semantics	Flags	Encoding	Opcode	Cond.
1	ADD	ADD Rd, Ra, Rb	$Rd \leftarrow Ra + Rb$	c,v,n,z	A	00100	-
2	ADDI	ADDI Rd, Ra, #imm5	$Rd \leftarrow Ra + imm5$	c,v,n,z	A	00101	-
3	ADDIB	ADDIB Rd, #imm8	$Rd \leftarrow Rd + imm8$	c,v,n,z	B	11000	-
4	ADC	ADC Rd, Ra, Rb	$Rd \leftarrow Ra + Rb + c$	c,v,n,z	A	00110	-
5	ADCI	ADCI Rd, Ra, #imm5	$Rd \leftarrow Ra + imm5 + c$	c,v,n,z	A	00111	-
6	NEG	NEG Rd, Ra	$Rd \leftarrow 0 - Ra$	c,v,n,z	A	01000	-
7	SUB	SUB Rd, Ra, Rb	$Rd \leftarrow Ra - Rb$	c,v,n,z	A	01001	-
8	SUBI	SUBI Rd, Ra, #imm5	$Rd \leftarrow Ra - imm5$	c,v,n,z	A	01010	-
9	SUBIB	SUBIB Rd, #imm8	$Rd \leftarrow Rd - imm8$	c,v,n,z	B	11001	-
10	SUC	SUC Rd, Ra, Rb	$Rd \leftarrow Ra - Rb - NOT\ c$	c,v,n,z	A	01011	-
11	SUCI	SUCI Rd, Ra, #imm5	$Rd \leftarrow Ra - imm5 - NOT\ c$	c,v,n,z	A	01100	-
12	CMP	CMP Ra, Rb	$Ra - Rb$	c,v,n,z	A	01101	-
13	CMPI	CMPI Ra, #imm5	$Ra - imm5$	c,v,n,z	A	01110	-
14	AND	AND Rd, Ra, Rb	$Rd \leftarrow Ra\ AND\ Rb$	z	A	10000	-
15	OR	OR Rd, Ra, Rb	$Rd \leftarrow Ra\ OR\ Rb$	z	A	10001	-
16	XOR	XOR Rd, Ra, Rb	$Rd \leftarrow Ra\ XOR\ Rb$	z	A	10010	-
17	NOT	NOT Rd, Ra	$Rd \leftarrow NOT\ Ra$	z	A	10011	-
18	NAND	NAND Rd, Ra, Rb	$Rd \leftarrow Ra\ NAND\ Rb$	z	A	10100	-
19	NOR	NOR Rd, Ra, Rb	$Rd \leftarrow Ra\ NOR\ Rb$	z	A	10110	-
20	LSL	LSL Rd, Ra, #imm4	$Rd \leftarrow Ra \ll imm4$	-	A	00001	-
21	LSR	LSR Rd, Ra, #imm4	$Rd \leftarrow Ra \gg imm4$	-	A	00010	-
22	ASR	ASR Rd, Ra, #imm4	$Rd \leftarrow Ra \ggg imm4$	-	A	00011	-
23	LDW	LDW Rd, [Ra, #imm5]	$Rd \leftarrow Mem[Ra + imm5]$	-	C	-	-
24	STW	SDW Rd, [Ra, #imm5]	$Mem[Ra + imm5] \leftarrow Rd$	-	C	-	-
25	LUI	LUI Rd, #imm8	$Rd[15:8] \leftarrow imm8$	-	B	11010	-
26	LLI	LLI Rd, #imm8	$Rd[7:0] \leftarrow imm8$	-	B	11011	-
27	BR	BR LABEL	$PC \leftarrow PC + imm8$	-	D	-	000
28	BNE	BNE LABEL	$(z==0)?\ PC \leftarrow PC + imm8$	-	D	-	110
29	BE	BE LABEL	$(z==1)?\ PC \leftarrow PC + imm8$	-	D	-	111
30	BLT	BLT LABEL	$(n\&\sim v\ OR\ \sim n\&v)?\ PC \leftarrow PC + imm8$	-	D	-	100
31	BGE	BGE LABEL	$(n\&v\ OR\ \sim n\&\sim v)?\ PC \leftarrow PC + imm8$	-	D	-	101
32	BWL	BWL LABEL	$LR \leftarrow PC + 1; PC \leftarrow PC + imm8$	-	D	-	011
33	RET	RET	$PC \leftarrow LR$	-	D	-	010
34	JMP	JMP Ra, #imm5	$PC \leftarrow Ra + imm5$	-	D	-	001
35	PUSH	PUSH {Ra, LR}	$Mem[SP] \leftarrow Ra; SP \leftarrow SP - 1;$	-	E	-	-
36	POP	POP {Ra, LR}	$SP \leftarrow SP + 1; Ra \leftarrow Mem[SP]$	-	E	-	-

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				Rd		Ra			Rb		X	X							
		Immediate										imm4/5										
B	Byte Immediate		Opcode				Rd		imm8													
C	Data Transfer		1	LS	1	0	1	Rd		Ra			imm5									
D	Control Transfer	Others	1	1	1	1	1	Cond.		imm8												
		Jump	1	1	1	1	1			Ra			imm5									
E	Stack Operations		1	1	1	0	0	U	L	X	Ra			X	X	X	X	X				

LS: 0 = Load Data, 1 = Store Data

U: 1 = PUSH, 0 = POP

L: 1 = Use Link, 0 = Don't use Link

Example Coding

Data Manipulation

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

1	ADD R5, R3, R4	$R5 \leftarrow R3 + R4$	13	CMPI R3, #9	$R3 - 9$
2	ADDI R5, R3, #9	$R5 \leftarrow R3 + 9$	14	AND R5, R3, R4	$R5 \leftarrow R3 \text{ AND } R4$
4	ADC R5, R3, R4	$R5 \leftarrow R3 + R4 + c$	15	OR R5, R3, R4	$R5 \leftarrow R3 \text{ OR } R4$
5	ADCI R5, R3, #9	$R5 \leftarrow R3 + 9 + c$	16	XOR R5, R3, R4	$R5 \leftarrow R3 \text{ XOR } R4$
6	NEG R5	$R5 \leftarrow 0 - R5$	17	NOT R5, R3	$R5 \leftarrow \text{NOT } R3$
7	SUB R5, R3, R4	$R5 \leftarrow R3 - R4$	18	NAND R5, R3, R4	$R5 \leftarrow R3 \text{ NAND } R4$
8	SUBI R5, R3, #9	$R5 \leftarrow R3 - 9$	19	NOR R5, R3, R4	$R5 \leftarrow R3 \text{ NOR } R4$
10	SUC R5, R3, R4	$R5 \leftarrow R3 - R4 - \text{NOT } c$	20	LSL R5, R3, #3	$R5 \leftarrow R3 \ll 3$
11	SUCI R5, R3, #9	$R5 \leftarrow R3 - 9 - \text{NOT } c$	21	LSR R5, R3, #3	$R5 \leftarrow R3 \gg 3$
12	CMP R3, R4	$R3 - R4$	22	ASR R5, R3, #3	$R5 \leftarrow R3 \ggg 3$

The value 'c' corresponds to the carry bit flag in the ALU from the previous calculation.

CMP, CMPI are comparison instructions for performing a subtraction without saving the result. The updated status flags can then be used for a conditional branch.

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	$R5 \leftarrow R5 + 150$
9	SUBIB R5, #150	$R5 \leftarrow R5 - 150$
25	LUI R5, #150	$R5[15:8] \leftarrow 150$
26	LLI R5, #150	$R5[7:0] \leftarrow 150$

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.

23	LDW R5, [R3, #imm5]	$R5 \leftarrow \text{Mem}[R3 + \text{imm5}]$
24	STW R5, [R3, #imm5]	$\text{Mem}[R3 + \text{imm5}] \leftarrow R5$

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 != – Conditionally branch if zero status flag (z) equals zero
- BE – Branch if = – Conditionally branch if zero status flag (z) equals one
- BLT – Branch if < – Conditionally branch if negative status flag (n) equals one
- BGE – Branch if ≥ – Conditionally branch if negative status flag (n) equals zero
- BWL – Branch with link – Unconditionally branch to stated location, saving PC to link register (LR)
- RET – Return – Unconditionally jump to the value stored in the link register (LR)
- JMP – Jump – Unconditionally jump to the location held in register Ra plus an 5-bit offset

Stack Operations

These operations are for popping or pushing either a general purpose register or the link register onto the system stack, useful for context saving when an interrupt occurs. PUSH pre-decrements stack pointer (SP) and POP post-increments stack pointer (SP) for a top-down growing stack. The 'U' bit indicates if a PUSH or POP operation is to be performed. If the 'L' bit is set, the link register value will be used instead of the value in register Ra.

Combined Branching & Stack Example

Below is an example showing how PUSH/POP operations and branches can be used to call a subroutine. ".sub" is a label used in assembly language to refer to a different line of code, it is converted to a relative address by an assembler. Here it is calculated as $4 + 5 = 9$, if the destination address was before the calling instruction the relative value would be negative.

	PUSH R1	:Save R1
	PUSH R2	:Save R2
	BWL .sub	:Call subroutine
	POP R2	:Restore R2
	POP R1	:Restore R1
	BR .end	:Branch to end of memory
.sub	PUSH LR	:Save Link Register
	...	:Subroutine does something
	POP LR	:Restore Link Register
	JMP	:Return to where subroutine was called
.end	BR .end	