**Instruction Set Summary**



**General Instruction Formatting**

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.



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.



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.



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 stack, useful for saving register values before or during a subroutine call. PUSH pre-decrements stack pointer (R7) and POP post-increments stack pointer (R7) 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 3 + 4 = 7, 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 |  |