Registers (repeats in **bold**):
GP registers: A, B, C; RL is also permitted unless otherwise noted.
Double-wide registers: R and PC; can also be addressed as RH, **RL** and PH, PL respectively.

- R is the return address register
- PC is the program counter register

Special registers: **RH**, **PL**, SP, IR

- SP is the stack pointer register
- IR is the interrupt register

Arithmetic registers: AA, AB, AC, AD
Status registers: F

- Contains the processor flags

"Dummy" registers: Z

- The zero register

Instructions:
MOV: Moves the value stored in arg1 to arg2
The only instruction with access to the relative addressing mode ($); MOV can write to and from memory addresses using this mode. Some forms (1, 2, and 3 listed below) also have access to special registers. When 16 bits is insufficient for a memory address, a register can be used as a high byte (for example, "$A 21FF" or "$A RL", though only the GP registers A, B, and C are permitted for this use. Z (the zero register) is also permitted, and will be used by default if the high byte register is omitted.
MOV can also be used to move double-width registers (R or PC) into two single-wide registers, or vice versa. For example, MOV R > A B will move the return address register's high byte to A, and its low byte to B.
Usage:

1) MOV ($)reg1 > ($)[regz]reg2
2) MOV ($)16L > ($)[regz]reg
3) MOV ($)reg > ($)[regz]16L
4) MOV reg > $24L
5) MOV $24L > reg
6) MOV ($)reg1 > ($)reg2 reg3
7) MOV ($)reg2 reg3 > ($)reg1
8) MOV largereg > reg1 reg2 OR
9) MOV reg1 reg2 > largereg OR
10) MOV 8L > reg

Notes:
1, 2, and 3 permit the use of special registers (RH, PL, SP, IR)
1, 2, and 3 also permit the use of the relative addressing mode
The regz field in 1, 2, and 3 may only be A, B, or C.

PSH: Pushes the given value to the stack
Increments the SP register and writes the given value to the top of the stack. If 1 double-width register or two single-width registers are provided, they will be pushed on the stack as a 32-bit word (incrementing SP by 2). Any registers (even arithmetic registers) can be used. This is the only instruction that can directly read the value of the F register.
Usage:

1) PSH reg
2) PSH 16L
3) PSH reg1 reg2
4) PSH largereg
5) PSH 8L

Notes:
ANY registers can be used

POP: Pops the top value off the stack and stores it in the given location
Decrements the SP register and populates the given register with the popped value. If a double-width register or 2 single-width registers are provided, a 32-bit word will be popped (SP will decrement by 2). Any registers (even arithmetic registers) can be used. This is the only instruction that can directly set the value of the F register.
Usage:

1) POP reg
2) POP reg1 reg2
3) POP largereg

Notes:
ANY registers can be used

JMP: Unconditional jump to the given location
Can jump to either an address or a label.
Usage:

1) JMP (reg) 16L
2) JMP (reg1) reg2
3) JMP 24L
4) JMP @Label

Notes:
Special registers are permitted.

JCD: Jump if condition is met
Conditions are as follows:

-

Usage: JCD cond, reg|16L

JRL: Relative jump
Special case of JCD; Equivalent to “JCD always”
Usage: JRL reg|16L

JSR: Jump to subroutine
Jumps to the specified location and populates the R register with the return address.
Usage:

1) JSR (reg) 16L
2) JSR (reg1) reg2
3) JSR 24L
4) JSR @Label

Notes:
Special registers are permitted.

RTS: Return from subroutine
Special case of JMP; Equivalent to “JMP RH RL”
Usage: RTS

ADD, ADC, SUB, SBC, AND, IOR, XOR: Binary arithmetic instructions
Performs the operator on arg1 and arg2, then stores the result in arg3. If arg3 is not specified, the result is instead stored in arg1.
Usage:

1) ADD reg1, reg2, reg3
2) ADD reg1, reg2
3) ADD reg1, 16L, reg3
4) ADD reg1, 16L

Notes:
Can use either GP registers or arithmetic registers

BSL, ROL, BSR, ROR: Unary arithmetic instructions
Performs the operation on arg1 and stores the result in arg2. If arg2 is not specified, the result is stored in arg1 instead.
Usage:

1) BSL reg1, reg2
2) BSL reg1

Notes: Can use either GP registers or arithmetic registers

SEF, CLF: Set flag, clear flag

Sets or clears the given flag. Valid values for flagName are as follows:

-

Usage:

1) SEF flagName
2) CLF flagName

Notes: affects the F status register

NOP: No operation
Waits one(?) cycle. Opcode is 0x00.
Usage: NOP

ITR: Forced interrupt
Triggers a forced interrupt and populates the low byte of the IR register with the 8-bit interrupt ID. The return PC value is pushed onto an internal stack. The high byte of IR is populated with flags regarding the interrupt.
Usage: ITR itrID

RTI: Return from interrupt
Returns from an interrupt routine. Return locations are read from an internal stack.
Usage: RTI