Field	**Bits of Instruction**	**Description**
Opcode	[15:12]	Operation code
Immediate enable / jump condition / subroutine select	[11]	Enables immediate operand, inverts jump conditions or enables subroutine instructions depending on opcode
Register A	[10:8]	Write register
Register B	[7:5]	Read register
Register C	[4:2]	Read register
Immediate operand	[7:0]	Immediate operand byte in instruction

Instruction	**Opcode**	**Immediate enable / jump condition / subroutine select**	**Instruction**	**Opcode**	**Immediate enable / jump condition / subroutine select**
MOV	0000	optional	RET	1111	1
ADD	0001	optional	JEQ	1001	0
SUB	0010	optional	JNE	1001	1
JMP	0011	optional	JCS	1010	0
LDR	0100	optional	JCC	1010	1
STR	0101	optional	JMI	1011	0
ADC	0110	optional	JPL	1011	1
SBC	0111	optional	JGE	1100	0
CMP	1000	optional	JLT	1100	1
PUSH	1110	0	JGT	1101	0
CALL	1110	1	JLE	1101	1
POP	1111	0			

Assembler Instructions

MOV Ra Rb	Ra = Rb
MOV Ra #n	Ra = #n
ADD Ra Rb Rc	Ra = Rb + Rc
ADD Ra #n	Ra = Ra + #n
SUB Ra Rb Rc	Ra = Rb - Rc
SUB Ra #n	Ra = Ra - #n
JMP #n	PC = PC + #n
LDR Ra Rc	Ra = mem[Rc]
LDR Ra #n	Ra = mem[#n]
STR Ra Rc	mem[Rc] = Ra
STR Ra #n	mem[#n] = Ra
ADC Ra Rb Rc	Ra = Rb + Rc + FlagC
ADC Ra #n	Ra = Ra + #n + FlagC
SBC Ra Rb Rc	Ra = Rb - Rc + (1 - FlagC)
SBC Ra #n	Ra = Ra - #n + (1 - FlagC)
CMP Rb Rc	Sets flags based on (Rb - Rc)
CMP Ra #n	Sets flags based on (Ra - #n)
PUSH Ra	Pushes Ra onto stack
POP Ra	Pops Ra off of stack
CALL #n	Calls subroutine PC = #n
RET	Returns from subroutine
JEQ #n	PC = PC + #n if flagZ = 1
JNE #n	PC = PC + #n if flagZ = 0
JCS #n	PC = PC + #n if flagC = 1
JCC #n	PC = PC + #n if flagC = 0
JMI #n	PC = PC + #n if flagN = 1
JPL #n	PC = PC + #n if flagN = 0
JGE #n	PC = PC + #n if flags ~(N⊕V) = 1
JLT #n	PC = PC + #n if flags (N⊕V) = 1
JGT #n	PC = PC + #n if flags ~(N⊕V).~Z
JLE #n	PC = PC + #n if flags (N⊕V) + Z = 1