

INSTRUCTION SET

Mnemonic	Opcode	Name	Function
MOV		Move r1, r2	r2:=r1
CMP		Complement r	r:=not(r)
CLR		Clear r	r:=0
INC		Increment r	r:=r PLUS 1
AND1		AND r3, r1, r2	r3:=r1 AND r2
ORR		OR r3, r1, r2	r3:=r1 OR r2
XOR1		XOR r3, r1, r2	r3:=r1 XOR r2
ADD		Add w/o carry r3, r1, r2	r3:=r1 PLUS r2
SUB		Subtract w/o borrow r3, r1, r2	r3:=r1 MINUS r2
ADC		Add w/ carry r3, r1, r2	r3:=r1 PLUS r2 PLUS CY
SBC		Subtract w/ borrow r3, r1, r2	r3:=r1 MINUS r2 MINUS BW
LSL		Logic shift left r, n	r:=LSL(r), n
LSR		Logic shift right r, n	r:=LSR(r), n
ASL		Algebraic shift left r, n	r:=ASL(r), n
ASR		Algebraic shift right r, n	r:=ASR(r), n
MUL		Unsigned multiply r3, r1, r2	r3:=r1 x r2 (16-bit result)
LDI		Load r, immediate	r:=data
LDR		Load r1, mem(r2)	r1=[mem(r2)]
LDD		Load double r1, mem(r2)	r1=[mem(r2)] (16-bit data)
STR		Store r1, mem(r2)	mem(r2):=(r1)
STD		Store double r1, mem(r2)	mem(r2):=(r1) (16-bit data)
CBR		Conditional branch (F), addr	PC:=addr if F=1, otherwise update PC
BRA		Branch, addr	PC:=addr
BRS		Branch to subroutine, addr	SPC:=PC, SPSW:=PSW, M=1
RTS		Return from subroutine	PC:=SPC, PSW:=SPSW, M=0
NOP		No operation	

PROGRAM CODE (entered through DIP Switches)

Location in Memory	Mnemonic	Word (hex value)	Expected Results
0000 (or F800)	LDI		Set 0 is used.
	data1		data1 is loaded into register A0.
	LDI		
	data2		data2 is loaded into register B0.
	AND		(A0) AND (B0) -> A1
	OR		(A0) OR (B0) -> 0B1
	STR		(A0) -> mem(0B1)
	MUL		(A0) * (B0) -> A2 and A3
	CLR		x"00" -> A3
	CBR on Z		Test Z flag to see if it is equal to 1; branch if so to
	address1		this address.

(At this point, the branch test succeeds and the program jumps to the provided address, address1.)

LDI	Fill up more Set 0 registers.
data3	data3 is loaded into register A4.
LDI	
data4	data4 is loaded into register B2.
SUB	(A4) - (B2) -> B4
ADD	(A4) + (B2) -> B3
CBR on S	Test S flag to see if it is equal to 1; branch if so to
address2	this address, address2.

(At this point, the branch test succeeds and the program jumps to the provided address, address2.)

ASL	ASL(A2, 3) -> A2
ASR	ASR(A0, 1) -> A0
BRS	Branch to subroutine at
address3	this address.

(At this point, a branch is called and the program jumps to the subroutine at address address3.)

LDI	Set 1 is used within the subroutine.
data4	data4 is loaded into register A0.
LDI	
data5	data5 is loaded into register B0.
LDR	[mem(B0)] -> B1
INC	(A0) + 1 -> A0
XOR	(A0) XOR (B0) -> A1
STR	(A1) -> mem(B0)
ADC	(A0) + (B0) + CY -> A2
SBC	(A2) - (B1) - BW -> A3
STR	(A0) -> mem(B0)
STR	(A2) -> mem(B1)
STR	(A3) -> mem(B0)
RTS	Return from subroutine.

(At this point, the program returns from the subroutine.)

LSL	Set 0 is used. LSL(A4, 4) -> A4
BRA	Branch to
address4	this address. (In this case, just steps ahead one address.)
LSR	LSR(A0, 5) -> A0
LDI	
data6	data6 is loaded into register B6.
STD	(A2) -> mem(B6), (A3) -> mem(B6 + 1)
LDD	[mem(B6)] -> A7, [mem(B6 + 1)] -> A0
MOV	(A2) -> B5
CMP	not(A7) -> A7
NOP	This is not a valid instruction, so nothing happens.
LDI	Addresses to be used for storing are loaded into Bank B.
data7	data7 -> 0B7
STR	(A0) -> mem(B7)
LDI	
data8	data8 -> B7
STR	(A1) -> mem(B7)
LDI	
data9	data9 -> B7
STR	(A2) -> mem(B7)
LDI	
data10	data10 -> B7
STR	(A3) -> mem(B7)
LDI	
data11	data11 -> B7
STR	(A4) -> mem(B7)
LDI	
data12	data12 -> B7
STR	(A7) -> mem(B7)
BRA	Indefinitely branch back to this address to keep program
address5	from “running away”.