Overview of Specifications:

word size : 16bit (Address and Data)
64kb memory (byte addressable)
8 general purpose registers
6 bit opcode + (3 bit reg identifier + 2 bit register mode bits) x 2 =
16 bit instruction (fixed size for instructions)

flag register :
compare A and B to set these flags (Ex: cmp A, B)
bit # 16 : Zero
bit # 15 : Equal
bit # 14 : Greater
bit # 13 : Lesser

register modes :
00 : reg has operand
01 : reg has pointer to operand
10 : immediate value (corresponding nibble in next instruction)
11 : <yet to be decided>
Note : 1st source operand uses 00, 01 & 10 modes
2nd source operand uses 00, 01 modes (same as destination)
Destination operand uses 00, 01 modes (same as 2nd source)
Also note that few instructions may not follow this pattern

Instruction opcode:
000000 : NEG (negation)
000001 : AND (logical AND)
000010 : XOR (logical XOR)
000011 : OR (logical OR)
000100 : LSR (logical right shift)
000101 : LSL/ASL (logical leftshift, equivalent to arithmetic leftshift)
000110 : ASR (arithmetic right shift)
000111 : TST (Test bit)
001000 : ROR (rotate right)
001001 : ROL (rotate left)
001010 : HLT (halt/stop)
001011 : <Not yet assigned>
001100 : MOVB (move byte)
001101 : MOV (move word)
001110 : CMP (compare)
001111 : JMP (jump; includes all sorts of jumps)
010000 : ADD (add)
010001 : SUB (sub)

Instruction Usage:

1. NEG : 000000
   1 operand (3 modes)
   1 destination (2 modes)

NEG op2, op1	op2 = ~op1	000000	xxx00	xxx00
NEG *op2, op1	*op2 = ~op1	000000	xxx01	xxx00
NEG op2, *op1	op2 = ~(*op1)	000000	xxx00	xxx01
NEG *op2, *op1	*op2 = ~(*op1)	000000	xxx01	xxx01
NEG op2, #val	op2 = ~(val) val = Immediate	000000 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
NEG *op2, #val	*op2 = ~(val) val = Immediate	000000 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

2. AND : 000001
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

AND op2, op1	op2 = op2 & op1	000001	xxx00	xxx00
AND *op2, op1	*op2 = *op2 & op1	000001	xxx01	xxx00
AND op2, *op1	op2 = op2 & *op1	000001	xxx00	xxx01
AND *op2, *op1	*op2 = *op2 & *op1	000001	xxx01	xxx01
AND op2, #val	op2 = op2 & val val = Immediate	000001 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
AND *op2, #val	*op2 = *op2 & val val = Immediate	000001 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

3. XOR : 000010
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

XOR op2, op1	op2 = op2 ^ op1	000010	xxx00	xxx00
XOR *op2, op1	*op2 = op2 ^ *op1	000010	xxx01	xxx00
XOR op2, *op1	op2 = *op2 ^ op1	000010	xxx00	xxx01
XOR *op2, *op1	*op2 = *op2 ^ *op1	000010	xxx01	xxx01
XOR op2, #val	op2 = op2 ^ val val = Immediate	000010 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
XOR *op2, #val	*op2 = *op2 ^ val val = Immediate	000010 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

4. OR : 000011
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

OR op2, op1	op2 = op1 | op2	000011	xxx00	xxx00
OR *op2, op1	*op2 = op1 | *op2	000011	xxx01	xxx00
OR op2, *op1	op2 = *op1 | op2	000011	xxx00	xxx01
OR *op2, *op1	*op2 = *op1 | *op2	000011	xxx01	xxx01
OR op2, #val	op2 = val | op2 val = Immediate	000011 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
OR *op2, #val	*op2 = val | *op2 val = Immediate	000011 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

5. LSR : 000100
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

LSR op2, op1	op2 = op2 >>> op1	000100	xxx00	xxx00
LSR *op2, op1	*op2 = *op2 >>> op1	000100	xxx01	xxx00
LSR op2, *op1	op2 = op2 >>> *op1	000100	xxx00	xxx01
LSR *op2, *op1	*op2 = *op2 >>>*op1	000100	xxx01	xxx01
LSR op2, #val	op2 = op2 >>> val val = Immediate	000100 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
LSR *op2, #val	*op2 = *op2 >>> val val = Immediate	000100 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

6. LSL : 000101
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

LSL op2, op1	op2 = op2 << op1	000101	xxx00	xxx00
LSL *op2, op1	*op2 = *op2 << op1	000101	xxx01	xxx00
LSL op2, *op1	op2 = op2 << *op1	000101	xxx00	xxx01
LSL *op2, *op1	*op2 = *op2 << *op1	000101	xxx01	xxx01
LSL op2, #val	op2 = op2 << val val = Immediate	000101 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
LSL *op2, #val	*op2 = *op2 << val val = Immediate	000101 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

7. ASR : 000110
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

ASR op2, op1	op2 = op2 >> op1	000110	xxx00	xxx00
ASR *op2, op1	*op2 = *op2 >> op1	000110	xxx01	xxx00
ASR op2, *op1	op2 = op2 >> *op1	000110	xxx00	xxx01
ASR *op2, *op1	*op2 = *op2 >> *op1	000110	xxx01	xxx01
ASR op2, #val	op2 = op2 >> val val = Immediate	000110 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
ASR *op2, #val	*op2 = *op2 >> val val = Immediate	000110 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

8. TST : 000111
   1 operand (2 modes for the only operand)
   Last 4 bits for bit# to test
   Z Flag updated. Rest are reset

TST op2, #0	Flag(Z) = !(bit 1)	000111	xxx00	x0000
TST op2, #1	Flag(Z) = !(bit 2)	000111	xxx00	x0001
TST op2, #2	Flag(Z) = !(bit 3)	000111	xxx00	x0010
TST op2, #3	Flag(Z) = !(bit 4)	000111	xxx00	x0011
TST op2, #4	Flag(Z) = !(bit 5)	000111	xxx00	x0100
TST op2, #5	Flag(Z) = !(bit 6)	000111	xxx00	x0101
TST op2, #6	Flag(Z) = !(bit 7)	000111	xxx00	x0110
TST op2, #7	Flag(Z) = !(bit 8)	000111	xxx00	x0111
TST op2, #8	Flag(Z) = !(bit 9)	000111	xxx00	x1000
TST op2, #9	Flag(Z) = !(bit 10)	000111	xxx00	x1001
TST op2, #10	Flag(Z) = !(bit 11)	000111	xxx00	x1010
TST op2, #11	Flag(Z) = !(bit 12)	000111	xxx00	x1011
TST op2, #12	Flag(Z) = !(bit 13)	000111	xxx00	x1100
TST op2, #13	Flag(Z) = !(bit 14)	000111	xxx00	x1101
TST op2, #14	Flag(Z) = !(bit 15)	000111	xxx00	x1110
TST op2, #15	Flag(Z) = !(bit 16)	000111	xxx00	x1111

   Similarly for 2nd mode (indirect mode) with 000111 xxx01 etc

9. ROR : 001000
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

ROR op2, op1	op2 = op2 -> op1	001000	xxx00	xxx00
ROR *op2, op1	*op2 = *op2 -> op1	001000	xxx01	xxx00
ROR op2, *op1	op2 = op2 -> *op1	001000	xxx00	xxx01
ROR *op2, *op1	*op2 = *op2 -> *op1	001000	xxx01	xxx01
ROR op2, #val	op2 = op2 -> val val = Immediate	001000 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
ROR *op2, #val	*op2 = *op2 -> val val = Immediate	001000 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

10. ROL : 001001
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

ROL op2, op1	op2 = op2 <- op1	001001	xxx00	xxx00
ROL *op2, op1	*op2 = *op2 <- op1	001001	xxx01	xxx00
ROL op2, *op1	op2 = op2 <- *op1	001001	xxx00	xxx01
ROL *op2, *op1	*op2 = *op2 <- *op1	001001	xxx01	xxx01
ROL op2, #val	op2 = op2 <- val val = Immediate	001001 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
ROL *op2, #val	*op2 = *op2 <- val val = Immediate	001001 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

11. HLT : 001010
   No operands

HLT	Halt	001010	xxxxx	xxxxx

13. MOVB : 001100
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

MOVB op2, op1	op2 = op1	001100	xxx00	xxx00
MOVB *op2, op1	*op2 = *op1	001100	xxx01	xxx00
MOVB op2, *op1	op2 = *op1	001100	xxx00	xxx01
MOVB *op2, *op1	*op2 = *op1	001100	xxx01	xxx01
MOVB op2, #val	op2 = val val = Immediate	001100 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
MOVB *op2, #val	*op2 = val val = Immediate	001100 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

14.MOV : 001101
   2 operands (3 modes for 1st, 2 modes for 2nd)
   1 destination (2 modes)

MOV op2, op1	op2 = op1	001101	xxx00	xxx00
MOV *op2, op1	*op2 = *op1	001101	xxx01	xxx00
MOV op2, *op1	op2 = *op1	001101	xxx00	xxx01
MOV *op2, *op1	*op2 = *op1	001101	xxx01	xxx01
MOV op2, #val	op2 = val val = Immediate	001101 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
MOV *op2, #val	*op2 = val val = Immediate	001101 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

15.CMP : 001110
   2 operands (3 modes for 1st, 2 modes for 2nd)
   Flag(Z,E,G,L) are updated. Reset are reset

   CMP op2, op1

   If op2 == op1 and both are zero,
         Flag(Z) = 1
         Flag(E) = 1
         Flag(G) = 0
         Flag(L) = 0

   If op2 == op1 and both are not zero,
         Flag(Z) = 0
         Flag(E) = 1
         Flag(G) = 0
         Flag(L) = 0

   If op2 > op1
         Flag(Z) = 0
         Flag(E) = 0
         Flag(G) = 1
         Flag(L) = 0

   If op2 < op1
         Flag(Z) = 0
         Flag(E) = 0
         Flag(G) = 0
         Flag(L) = 1

16. JMP : 001111
   1 operand - Jump Destination (jmpdest)
   NOTE: JMP uses special register modes for its only operand
   Last 5 bits of instruction control JMP behaviour

   JMP Special Register Modes:
   00 : Register has absolute address to jump to
   01 : Register has offset to be added to PC
   10 : Immediate value has absolute address to jump to
   11 : Immediate value has offset to be added to PC

   If bit #5 is set, the jump is always taken; ignoring bits 1-4
   If bit #5 is not set, the jump is taken corresponding to the Flag bits.
   The last 4 bits correspond to the Flag(Z,E,G,L) bits.
   Eg1 : Jump if greater than must check if Flag(G) is set. For this, use the encoding 0010 for the last nibble.
   Eg2 : Jump if not equal must check is Flag(Z,E) is not set. For this, use encoding 0011. The jump is taken if either Flag(G) or Flag(L) is set which serves the purpose.
   Note that it checks if corresponding bits are set only. It does not check whether a bit is set or not set.

   Reference Table :

JMP	001111	xxxMM	1xxxx
JZ	001111	xxxMM	01000
JNZ	001111	xxxMM	00111
JG	001111	xxxMM	00010
JL	001111	xxxMM	00001
JE	001111	xxxMM	01100
JNE	001111	xxxMM	00011
JGE	001111	xxxMM	01110
JLE	001111	xxxMM	01101

   'MM' represents special register mode for JMP Instruction.
   If 'val' is the value present in reg xxx or immediate value (next word)
   And if jumpdest is the destination to jump to :
   00 => jumpdest = reg(val)
   01 => jumpdest = pc + reg(val)
   10 => jumpdest = #val
   11 => jumpdest = pc + #val

17. ADD : 010000
    2 operands (3 modes for 1st, 2 modes for 2nd)
    1 destination (2 modes)

ADD op2, op1	op2 = op2 + op1	010000	xxx00	xxx00
ADD *op2, op1	*op2 = *op2 + op1	010000	xxx01	xxx00
ADD op2, *op1	op2 = op2 + *op1	010000	xxx00	xxx01
ADD *op2, *op1	*op2 = *op2 + *op1	010000	xxx01	xxx01
ADD op2, #val	op2 = op2 + val val = Immediate	010000 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
ADD *op2, #val	*op2 = *op2 + val val = Immediate	010000 xxxxxx	xxx01 xxxxx	xxx10 xxxxx

18. SUB : 010001
    2 operands (3 modes for 1st, 2 modes for 2nd)
    1 destination (2 modes)

SUB op2, op1	op2 = op2 - op1	010001	xxx00	xxx00
SUB *op2, op1	*op2 = *op2 - op1	010001	xxx01	xxx00
SUB op2, *op1	op2 = op2 - *op1	010001	xxx00	xxx01
SUB *op2, *op1	*op2 = *op2 - *op1	010001	xxx01	xxx01
SUB op2, #val	op2 = op2 - val val = Immediate	010001 xxxxxx	xxx00 xxxxx	xxx10 xxxxx
SUB *op2, #val	*op2 = *op2 - val val = Immediate	010001 xxxxxx	xxx01 xxxxx	xxx10 xxxxx