Contents

[1 INTRODUCTION 3](#_Toc154936647)

[1.1 FEATURES OF THE BES8 3](#_Toc154936648)

[2 FUNCTIONAL DESRIPTION 3](#_Toc154936649)

[2.1 Arithmetic and Logic Unit (ALU) 3](#_Toc154936650)

[2.4 General Purpose Register (AX, BX, CX, DX, ZX) 3](#_Toc154936651)

[2.5 Index Registers (X, Y) 3](#_Toc154936652)

[2.6 General Purpose Register 32 bit (EAX, EBX) 4](#_Toc154936653)

[2.7 Flags Register (F) 4](#_Toc154936654)

[2.8 Program Counter Register (PC) 4](#_Toc154936655)

[2.9 Stack Pointer Register (S) 4](#_Toc154936656)

[3 Addressing Modes 5](#_Toc154936657)

[3.1 ABSOLUTE A 5](#_Toc154936658)

[3.2 ABSOLUTE INDEXED WITH INDEX RRG A, I 5](#_Toc154936659)

[3.2 IMMEDITATE # 5](#_Toc154936660)

[3.3 IMPLIED I 5](#_Toc154936661)

[3.4 REGISTER 5](#_Toc154936662)

[3.5 32-bit IMMEDITATE 5](#_Toc154936663)

[4. OPERATION TABLES 6](#_Toc154936664)

# 1 INTRODUCTION

## FEATURES OF THE BES8

* 16-bit data bus
* 16-bit address bus provides access to 65,536 bytes of memory space
* 16-bit ALU, AX, BX, CX, DX, Z, Stack Pointer, Index Registers, Memory Bank Register
* 32-bit Program Counter
* 32-bit EAX, EBX
* 34 instructions
* 6 Argument Modes
* 59 Operation Codes (OpCodes)
* Op to 511 ports

# 2 FUNCTIONAL DESRIPTION

## 2.1 Arithmetic and Logic Unit (ALU)

All arithmetic and logic operations take place within the ALU. In addition to data operations, the ALU also calculates the effective address for relative and indexed addressing modes. The result of a data operation is stored in either memory or an internal register. Carry, Negative, Overflow and Zero flags are updated following the ALU data operation.

## 2.4 General Purpose Register (AX, BX, CX, DX, ZX)

The AX, BX, CX and DX registers are 16-bit general purpose registers which holds one of the operands and the result of arithmetic and logical operations. All of these registers can be split into two 8-bit registers

## 2.5 Index Registers (X, Y)

There are two 16-bit Index Registers (X, Y) which may be used as general purpose registers or to provide an index value for calculation of the effective address. When executing an instruction with indexed addressing, the microprocessor fetches the OpCode and the base address, and then modifies the address by adding the Index Register contents to the address prior to performing the desired operation.

## 2.6 General Purpose Register 32 bit (EAX, EBX)

The EAX, EBX registers are 32-bit general purpose registers which holds one of the operands and the result of arithmetic and logical operations.

## 2.7 Flags Register (F)

The 16-bit Processor Status Register (F) contains status flags and mode select bits. The Carry (C), Negative (N), Overflow (V) and Zero (Z) status flags serve to report the status of ALU operations. These status flags are tested with Conditional Jump instructions.

## 2.8 Program Counter Register (PC)

The 32-bit Program Counter Register (PC) provides the addresses which are used to step the microprocessor through sequential program instructions. This register is incremented each time an instruction or operand is fetched from program memory.

## 2.9 Stack Pointer Register (S)

The Stack Pointer Register (S) is a 16-bit register which is used to indicate the next available location in the stack memory area. It serves as the effective address in stack addressing modes as well as subroutine and interrupt processing.

# 3 Addressing Modes

ADH = address high

ADL = address low

REGH = register high

REGL = register low

## 3.1 ABSOLUTE A

With Absolute addressing the second and third byte is the 16-bit address.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| OpCode | 0001 | ADH | ADL |

byte:

Instruction:

## 3.2 ABSOLUTE INDEXED WITH INDEX RRG A, I

With absolute indexed addressing you can use one of the index registers to and an offset to the absolute address

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| OpCode | 0011 | ADH | ADL | REGH | REGL |

byte:

Instruction:

## 3.2 IMMEDITATE #

With immediate addressing the data is in the instruction itself.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| OpCode | 0011 | Operand-H | Operand-L |

byte:

Instruction:

## 3.3 IMPLIED I

With implied addressing the instruction is only two bytes

|  |  |
| --- | --- |
| 0 | 1 |
| OpCode | 1111 |

Byte:

Instruction:

## 3.4 REGISTER

## 3.5 32-bit IMMEDITATE

With immediate addressing the data is in the instruction itself.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| OpCode | 1000 | Op-HH | Op-HL | Op-LH | Op-LL |

Byte:

Instruction:

**Addressing Mode Table**

# 4. OPERATION TABLES

|  |  |  |
| --- | --- | --- |
| 00 | MOV | MOVes an immediate/register/address into a register |
| 01 | STR | DES |
| 02 | LOR | DES |
| 03 | PUSH | DES |
| 04 | POP | DES |
| 05 | ADD | DES |
| 06 | STI | DES |
| 07 | SUB | DES |
| 08 | AND | DES |
| 09 | OR | DES |
| 10 | NOR | DES |
| 11 | CMP | DES |
| 12 | JNE | Jump Not if Equals (E=1) |
| 13 | INT | DES |
| 14 | JMP | JuMP to new location |
| 15 | JME | JuMp if Equals (E=1) |
| 16 | HELT | DES |
| 17 | CALL | DES |
| 18 | RTS | DES |
| 19 | INC | INCrement a register or memory |
| 20 | DEC | DECrement a register or memory |
| 21 | OUTP | OUTputs to a Port |
| 22 | INP | INputs from a Port |
| 23 | NOP | DES |
| 24 | SEF | DES |
| 25 | CLF | DES |
| 26 | JMZ | DES |
| 27 | JNZ | DES |
| 28 | JML | DES |
| 29 | JMG | DES |
| 30 | JMC | DES |
| 31 | JNC | DES |
| 32 | MUL |  |
| 33 | DIV |  |
| 34 | NOT |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | MOV | STR | LOR | PUSH | POP | ADD | STI | SUB | AND | OR | NOR | CMP | JNE | INT | JMP | JME |
| 1 | HALT | CALL | RTS | INC | DEC | OUTB | INB | NOP | SEF | CLF | JMZ | JNZ | JML | JMG | JMC | JNC |
| 2 | MUL | DIV | NOT |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| F |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Mnemomic | Operation  # Immediate Data  & AND | OR |! NOR  ! NOT  - SUB + ADD  R Register  V Value | Implied | Immediate # | Absolute | Register | Indexed | 32-bit Immediate | Flags Register (F)  \*Use Defined | | | | | | | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| L | E | Z | C | I | H | S | P |  |  |  |  |  |  |  |  |
| **MOV** | R = V |  | 0000 | 0001 | 0002 | 0003 | 0008 | . | . | Z | . | . | . | S | P | . | . | . | . | . | . | . | . |
| **STR** | M = R |  |  | 0101 |  | 0103 | 0108 | . | . | Z | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **LOR** | R = M |  |  | 0201 |  | 0203 | 0208 | . | . | Z | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **PUSH** | S = V S-- |  | 0300 |  | 0302 | 0303 |  | . | . | . | C | . | . | . | . | . | . | . | . | . | . | . | . |
| **POP** | S++ R = S |  |  |  | 0402 |  |  | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **ADD** | R = R + V |  | 0500 | 0501 | 0502 |  |  | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **STI** | M = # |  | 0600 |  |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **SUB** | R = R - V |  | 0700 | 0701 | 0702 |  |  | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **AND** | R = R & V |  | 0800 | 0801 | 0802 |  |  | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **OR** | R = R | V |  | 0900 | 0901 | 0902 |  |  | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **NOR** | R = R |! V |  | 0A00 | 0A01 | 0A02 |  |  | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **CMP** | R - V |  | 0B00 | 0B01 | 0B02 |  |  | L | E | Z | . | . | . | S | P | . | . | . | . | . | . | . | . |
| **JNE** | Jump if E = 0 |  |  | 0C01 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **INT** | INT | 0D0F |  |  |  |  |  | . | . | . | . | 0 | H | . | . | . | . | . | . | . | . | . | . |
| **JMP** | Jumps |  |  | 0E01 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **JME** | Jump if E = 1 |  |  | 0F01 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **HALT** | Stops the CPU | 100F | 1000 |  |  |  |  | . | . | . | . | 1 | . | . | . | . | . | . | . | . | . | . | . |
| **CALL** | S-- = PC = V |  |  | 1101 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **RTS** | PC = S++ | 120F |  |  |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **INC** | V++ |  |  | 1301 | 1302 |  |  | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **DEC** | V-- |  |  | 1401 | 1402 |  |  | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **OUTB** | P = V |  | 1500 |  | 1502 |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **INB** | R = P |  |  |  | 1602 |  |  | . | . | Z | . | . | . | S | P | . | . | . | . | . | . | . | . |
| **NOP** | NOP | 170F |  |  |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **SEF** | Set a Flag |  | 1800 |  |  |  |  | L | E | Z | C | I | . | S | P | . | . | . | . | . | . | . | . |
| **CLF** | Clears a Falg |  | 1900 |  |  |  |  | L | E | Z | C | I | . | S | P | . | . | . | . | . | . | . | . |
| **JMZ** | Jump if Z = 1 |  |  | 1A01 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **JNZ** | Jump if Z = 0 |  |  | 1B01 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **JML** | Jump if L = 1 |  |  | 1C01 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **JMG** | Jump if L = 0 |  |  | 1D01 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **JMC** | Jump if C = 1 |  |  | 1E01 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **JNC** | Jump if C = 0 |  |  | 1F01 |  |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **MUL** | R = R \* V |  | 2000 | 2001 | 2002 |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **DIV** | R = R / V |  | 2100 | 2101 | 2102 |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **NOT** | R = ! R |  |  |  | 2202 |  |  | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **ADDL** | R32 = R32 + V32 |  |  |  | 2302 |  | 2308 | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **SUBL** | R32 = R32 - V32 |  |  |  | 2402 |  | 2408 | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **ANDL** | R32 = R32 & V32 |  |  |  | 2502 |  | 2508 | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **ORL** | R32 = R32 | V32 |  |  |  | 2602 |  | 2608 | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **NORL** | R32 = R32 !| V32 |  |  |  | 2702 |  | 2708 | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **NOTL** | R32 = ! R32 |  |  |  | 2802 |  | 2808 | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |
| **JMPL** | PC = R32 |  |  |  | 2902 |  |  | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| **PUSHL** | S = V32 S-- |  |  |  | 3002 |  | 3008 | . | . | . | C | . | . | . | . | . | . | . | . | . | . | . | . |
| **POPL** | S++ R32 = S |  |  |  | 3102 |  | 3108 | . | . | Z | C | . | . | S | P | . | . | . | . | . | . | . | . |