Department of Electrical Engineering

Faculty Member: **Dated:**

Course/Section: **Semester:**

Computer Organization and Assembly Language (CS235)

Lab

Programming: Addressing modes (part 1)

Group.

Name	Reg. no.	Report Marks / 5	Viva Marks / 10	Total/15

Objective

Upon completion of this lab, you will be able to describe and execute the addressing modes of the 80386 CPU by using the 32 bit MICROPROCESSOR circuit board.

Pre Lab

Introduction to Registers (revision)

Processor operations mostly involve processing data. This data can be stored in memory and accessed from thereon. However, reading data from and storing data into memory slows down the processor, as it involves complicated processes of sending the data request across the control bus and into the memory storage unit and getting the data through the same channel.

To speed up the processor operations, the processor includes some internal memory storage locations, called **registers**.

The registers store data elements for processing without having to access the memory. A limited number of registers are built into the processor chip.

Processor Registers

There are ten 32-bit and six 16-bit processor registers in IA-32 architecture. The registers are grouped into three categories −

- General registers,
- Control registers, and
- Segment registers.

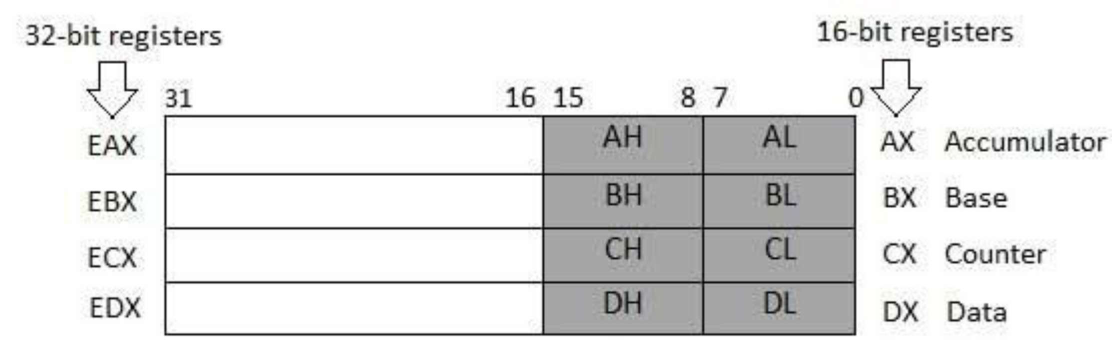
The general registers are further divided into the following groups −

- Data registers,
- Pointer registers, and
- Index registers.

Data Registers

Four 32-bit data registers are used for arithmetic, logical, and other operations. These 32-bit registers can be used in three ways −

- As complete 32-bit data registers: EAX, EBX, ECX, EDX.
- Lower halves of the 32-bit registers can be used as four 16-bit data registers: AX, BX, CX and DX.
- Lower and higher halves of the above-mentioned four 16-bit registers can be used as eight 8-bit data registers: AH, AL, BH, BL, CH, CL, DH, and DL.



Some of these data registers have specific use in arithmetical operations.

AX is the primary accumulator; it is used in input/output and most arithmetic instructions. For example, in multiplication operation, one operand is stored in EAX or AX or AL register according to the size of the operand.

BX is known as the base register, as it could be used in indexed addressing.

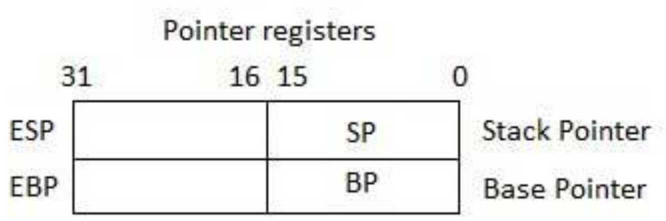
CX is known as the count register, as the ECX, CX registers store the loop count in iterative operations.

DX is known as the data register. It is also used in input/output operations. It is also used with AX register along with DX for multiply and divide operations involving large values.

Pointer Registers

The pointer registers are 32-bit EIP, ESP, and EBP registers and corresponding 16-bit right portions IP, SP, and BP. There are three categories of pointer registers −

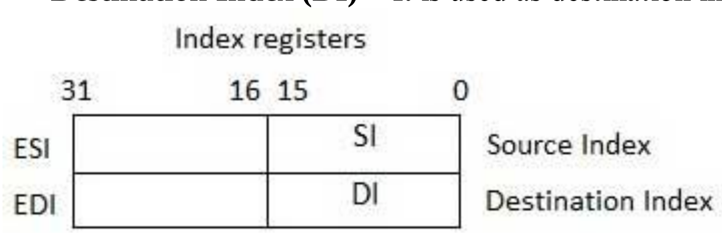
- **Instruction Pointer (IP)** − The 16-bit IP register stores the offset address of the next instruction to be executed. IP in association with the CS register (as CS:IP) gives the complete address of the current instruction in the code segment.
- **Stack Pointer (SP)** − The 16-bit SP register provides the offset value within the program stack. SP in association with the SS register (SS:SP) refers to be current position of data or address within the program stack.
- **Base Pointer (BP)** − The 16-bit BP register mainly helps in referencing the parameter variables passed to a subroutine. The address in SS register is combined with the offset in BP to get the location of the parameter. BP can also be combined with DI and SI as base register for special addressing.



Index Registers

The 32-bit index registers, ESI and EDI, and their 16-bit rightmost portions. SI and DI, are used for indexed addressing and sometimes used in addition and subtraction. There are two sets of index pointers −

- **Source Index (SI)** − It is used as source index for string operations.
- **Destination Index (DI)** − It is used as destination index for string operations.



Control Registers

The 32-bit instruction pointer register and the 32-bit flags register combined are considered as the control registers.

Many instructions involve comparisons and mathematical calculations and change the status of the flags and some other conditional instructions test the value of these status flags to take the control flow to other location.

The common flag bits are:

- **Overflow Flag (OF)** − It indicates the overflow of a high-order bit (leftmost bit) of data after a signed arithmetic operation.
- **Direction Flag (DF)** − It determines left or right direction for moving or comparing string data. When the DF value is 0, the string operation takes left-to-right direction and when the value is set to 1, the string operation takes right-to-left direction.
- **Interrupt Flag (IF)** − It determines whether the external interrupts like keyboard entry, etc., are to be ignored or processed. It disables the external interrupt when the value is 0 and enables interrupts when set to 1.
- **Trap Flag (TF)** − It allows setting the operation of the processor in single-step mode. The DEBUG program we used sets the trap flag, so we could step through the execution one instruction at a time.
- **Sign Flag (SF)** − It shows the sign of the result of an arithmetic operation. This flag is set according to the sign of a data item following the arithmetic operation. The sign is indicated by the high-order of leftmost bit. A positive result clears the value of SF to 0 and negative result sets it to 1.
- **Zero Flag (ZF)** − It indicates the result of an arithmetic or comparison operation. A nonzero result clears the zero flag to 0, and a zero result sets it to 1.
- **Auxiliary Carry Flag (AF)** − It contains the carry from bit 3 to bit 4 following an arithmetic operation; used for specialized arithmetic. The AF is set when a 1-byte arithmetic operation causes a carry from bit 3 into bit 4.
- **Parity Flag (PF)** − It indicates the total number of 1-bits in the result obtained from an arithmetic operation. An even number of 1-bits clears the parity flag to 0 and an odd number of 1-bits sets the parity flag to 1.
- **Carry Flag (CF)** − It contains the carry of 0 or 1 from a high-order bit (leftmost) after an arithmetic operation. It also stores the contents of last bit of a *shift* or *rotate* operation.

The following table indicates the position of flag bits in the 16-bit Flags register:

Flag:					O	D	I	T	S	Z		A		P		C
Bit no:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Segment Registers

Segments are specific areas defined in a program for containing data, code and stack. There are three main segments −

- **Code Segment** − It contains all the instructions to be executed. A 16-bit Code Segment register or CS register stores the starting address of the code segment.
- **Data Segment** − It contains data, constants and work areas. A 16-bit Data Segment register or DS register stores the starting address of the data segment.
- **Stack Segment** − It contains data and return addresses of procedures or subroutines. It is implemented as a 'stack' data structure. The Stack Segment register or SS register stores the starting address of the stack.

Apart from the DS, CS and SS registers, there are other extra segment registers - ES (extra segment), FS and GS, which provide additional segments for storing data.

In assembly programming, a program needs to access the memory locations. All memory locations within a segment are relative to the starting address of the segment. A segment begins in an address evenly divisible by 16 or hexadecimal 10. So, the rightmost hex digit in all such memory addresses is 0, which is not generally stored in the segment registers.

The segment registers stores the starting addresses of a segment. To get the exact location of data or instruction within a segment, an offset value (or displacement) is required. To reference any memory location in a segment, the processor combines the segment address in the segment register with the offset value of the location.

Variables in Assembly language

Intrinsic Data Types

BYTE; SBYTE	8-bit unsigned integer; 8-bit signed integer
WORD, SWORD	16-bit unsigned & signed integer
DWORD, SDWORD	32-bit unsigned & signed integer
QWORD	64-bit integer
TBYTE	80-bit integer

Lab Task

Answer the following questions after completing the lab experiments (Unit 6)

1. Where is the location of the instruction pointer, segment, general purpose, and the flags registers?

2. In the real mode, what does the physical address of the next instruction code equal?

3. In the instruction MOV CX, ABE5H, what addressing mode is used and after execution, what will the CS register contain?

4. For 16-bit and 32-bit addressing, what value(s) can you use to compose an effective address (EA)?

5. For 32-bit addressing, the base value is in what register(s)? Answer the question by considering the figure 1 below.

REGISTERS, SCALE FACTORS, AND DISPLACEMENTS FOR 16- AND 32-BIT ADDRESSING		
	16-BIT ADDRESSING	32-BIT ADDRESSING
DISPLACEMENT SIZE	0, 8, 16 BITS	0, 8, 32 BITS
BASE VALUE REGISTER	BX, BP	ANY 32-BIT GP REGISTER
INDEX VALUE REGISTER	SI, DI	ANY 32-BIT REGISTER EXCEPT ESP
SCALE FACTOR	NONE	1, 2, 4, 8

Figure 1 Question 5

Department of Electrical Engineering

Faculty Member: **Dated:**

Course/Section: **Semester:**

Computer Organization and Assembly Language (CS235)

Lab

Programming: Addressing modes (part 2)

Group.

Name	Reg. no.	Report Marks / 5	Viva Marks / 10	Total/15

Objective

Upon completion of this lab, you will be able to describe and execute the addressing modes of the 80386 CPU by using the 32 bit MICROPROCESSOR circuit board.

Lab Tasks

6. When the base value is in the BP register (see figure 2), what is the default segment for data?

SEGMENT REGISTER SELECTION RULES

TYPE OF MEMORY REFERENCE	DEFAULT SEGMENT	ALTERNATE SEGMENT	OFFSET
INSTRUCTION CODE FETCH	CS	NONE	IP
STACK OPERATION	SS	NONE	SP
DATA (WHEN EAX,EBX,ECX EDX,ESI,OR EDI USED FOR BASE VALUE)	DS	CS,SS,ES, FS,GS	EFFECTIVE ADDRESS
DATA (WHEN EBP OR ESP USED FOR BASE VALUE)	SS	CS,DS,ES, FS,GS	EFFECTIVE ADDRESS

Figure 2 Question 6

7. The instruction MOV EDX, [DI+BX+0030H] uses what type of addressing mode?

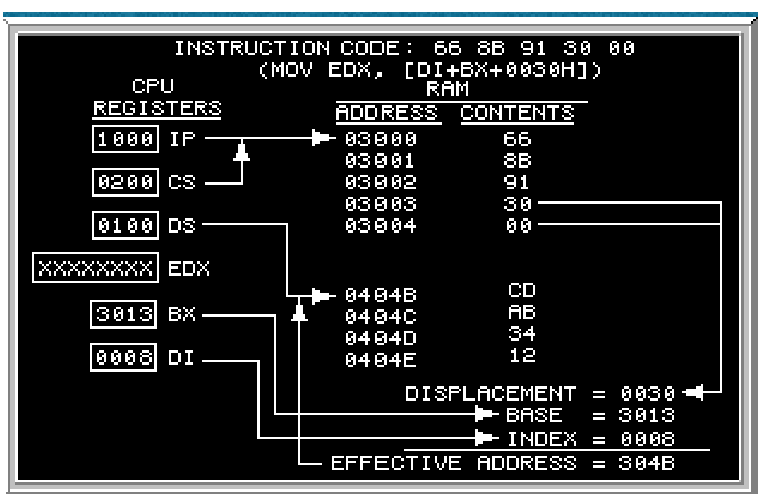


Figure 3 Question 7

8. What addressing mode can be used only for 32-bit addressing?

9. For 32 bit addressing using a scaled index, what values can the scale factor have?

REGISTERS, SCALE FACTORS, AND DISPLACEMENTS FOR 16- AND 32-BIT ADDRESSING		
	16-BIT ADDRESSING	32-BIT ADDRESSING
DISPLACEMENT SIZE	0, 8, 16 BITS	0, 8, 32 BITS
BASE VALUE REGISTER	BX, BP	ANY 32-BIT GP REGISTER
INDEX VALUE REGISTER	SI, DI	ANY 32-BIT REGISTER EXCEPT ESP
SCALE FACTOR	NONE	1, 2, 4, 8

Figure 4 Question 9

10. In the real mode, what instruction code hexadecimal prefix must you use when applying a scaled index in the EA calculation?