Experiments No:8

Title: Find factorial of a given integer number.

Problem Statement: Write X86/64 ALP to perform non-overlapped block transfer without string specific instructions. Block containing data can be defined in the data segment.

Objective:

- Understand the memory Addressing
- Understand the localization of Data

Outcomes: On completion of this practical ,students will be able to

C218.1: Understand and apply various addressing modes and instruction set to implement assembly language programs

Hardware Requirement: NA

Software Requirement: OS: Ubuntu Assembler: NASM version 2.10.07 Linker: ld

Theory Contents :

2.3.1 Registers

Registers are places in the CPU where a number can be stored and manipulated. There are three sizes of registers: 8-bit, 16-bit and on 386 and above 32-bit. There are four different types of registers:

1. **General Purpose Registers,**
2. **Segment Registers,**
3. **Index Registers,**
4. **Stack Registers.**

2.3.1.1 General Registers

Following are the registers that are used for general purposes in 8086

AX accumulator (16 bit)

AH accumulator high-order byte (8 bit)

AL accumulator low-order byte (8 bit)

BX accumulator (16 bit)

BH accumulator high-order byte (8 bit)

BL accumulator low-order byte (8 bit)

CX count and accumulator (16 bit)

CH count high order byte (8 bit)

CL count low order byte (8 bit)

DX data and I/O address (16 bit)

DH data high order byte (8 bit)

DL data low order byte (8 bit)

2.3.1.2 Segment Registers –

These registers are used to calculate 20 bit address from 16 bit registers.

CS code segment (16 bit)

DS data segment (16 bit)

SS stack segment (16 bit)

ES extra segment (16 bit)

2.3.1.3 Index Registers –

These registers are used with the string instructions.

DI destination index (16 bi

SI source index (16 bit)

2.3.1.4 Pointers –

These registers are used with the segment register to obtain 20 bit addresses

SP stack pointer (16 bit) BP base pointer (16 bit)

IP instruction pointer (16 bit)

CS, Code Segment

Used to "point" to Instructions Determines a Memory Address (along with IP) Segmented Address written as CS:IP

DS, Data Segment

Used to "point" to Data Determines Memory Address (along with other registers) ES, Extra Segment allows 2 Data Address Registers

SS, Stack Segment

Used to "point" to Data in Stack Structure (LIFO) Used with SP or BP

SS: SP or SS:BP are valid Segmented

Addresses IP, Instruction Pointer

Used to "point" to Instructions Determines a Memory Address (along with CS) Segmented Address written as CS:IP

SI, Source Index; DI, Destination Index

Used to "point" to Data Determines Memory Address (along with other registers) DS, ES commonly used

SP, Stack Pointer; BP, Base Pointer

Used to "point" to Data in Stack Structure (LIFO) Used with SS SS:SP or SS:BP are valid Segmented Address

2.3.2 Memory Address Calculations

The 8086 uses a 20 bit address but the registers are only sixteen bit. To derive twenty bit addresses from the registers two registers are Combined every memory reference uses one of the four segment registers plus and offset abd/or a base pointer and/or a index register. The segment register is multiplied by sixteen (shifted to the left four bits) and added to the sixteen bit result of the offset calculation.

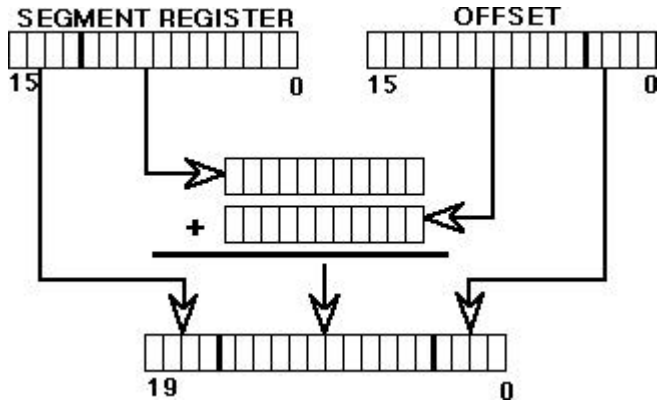


Figure 1Diagrammatic Representation of Address calculation

The 8086 provides four segment registers for address calculations. Each segment register is assigned a different task. The code segment register is always used with the instruction Pointer (also called the program counter) to point to the instruction that is to be executed next. The stack segment register is always used with the stack pointer to point to the last value pushed onto the stack. The extra segment is general purpose segment register. The data segment register is the default register to calculate data operations, this can be over ridden by specifying the segment register. For example *mov ax,var1* would use the offset var1 and the data segment to calculate the memory reference but *mov ax,ss:var1* would use the offset var1 and the stack segment register

to calculate the memory reference.

The offset can be calculated in a number of ways. Their are three elements that can make up an offset. The first element is a base register, this can be one of the BX of BP registers (the BP register defaults to the stack segment). The second element is one of the index register, SI or DI. The third element is a displacement. A displacement can be a numerical value or an offset to a label. An offset can contain one to three of these elements, making a total of sixteen possibilities.

BX SI
or + or +
Displacement BP
DI
(base) (index)

The offset to a label in calculated using the assembler directive OFFSET. This directive makes the assembler calculattehe distant from the start of the segment that the label resides in to the label.

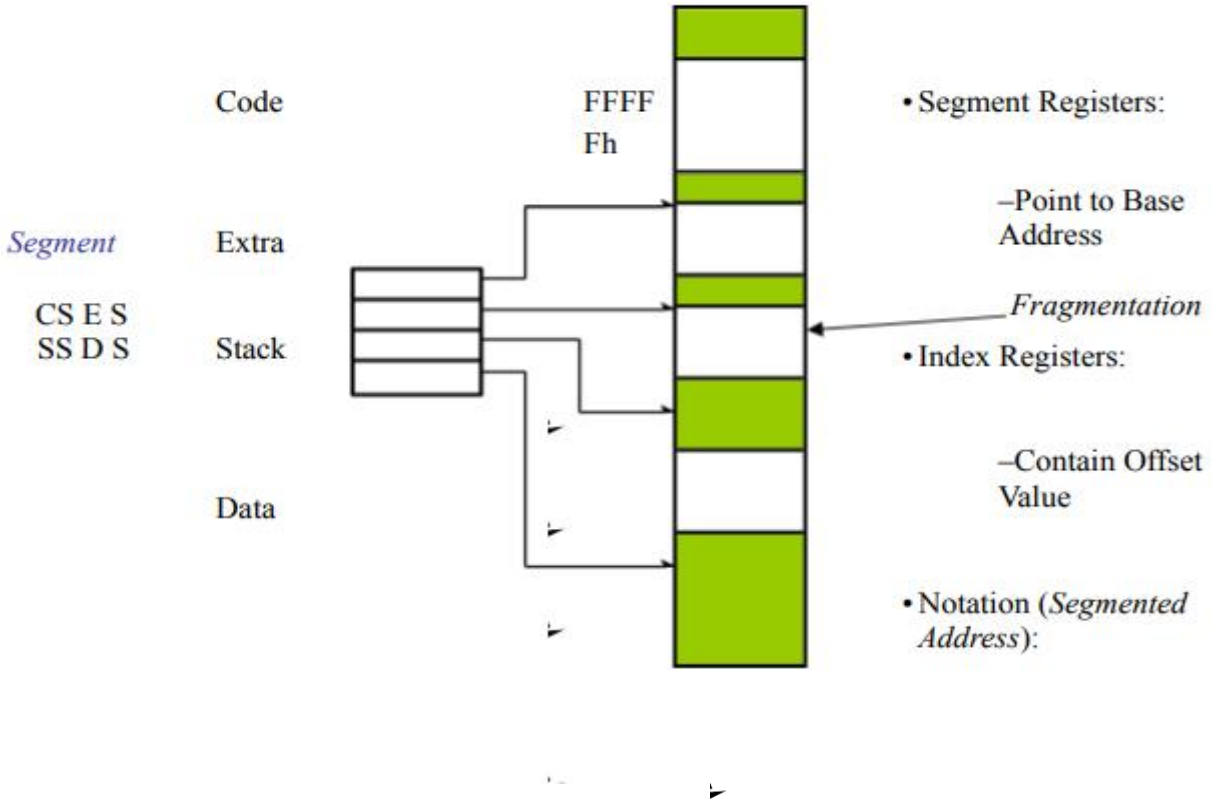
2.3.3 Memory Segmentation

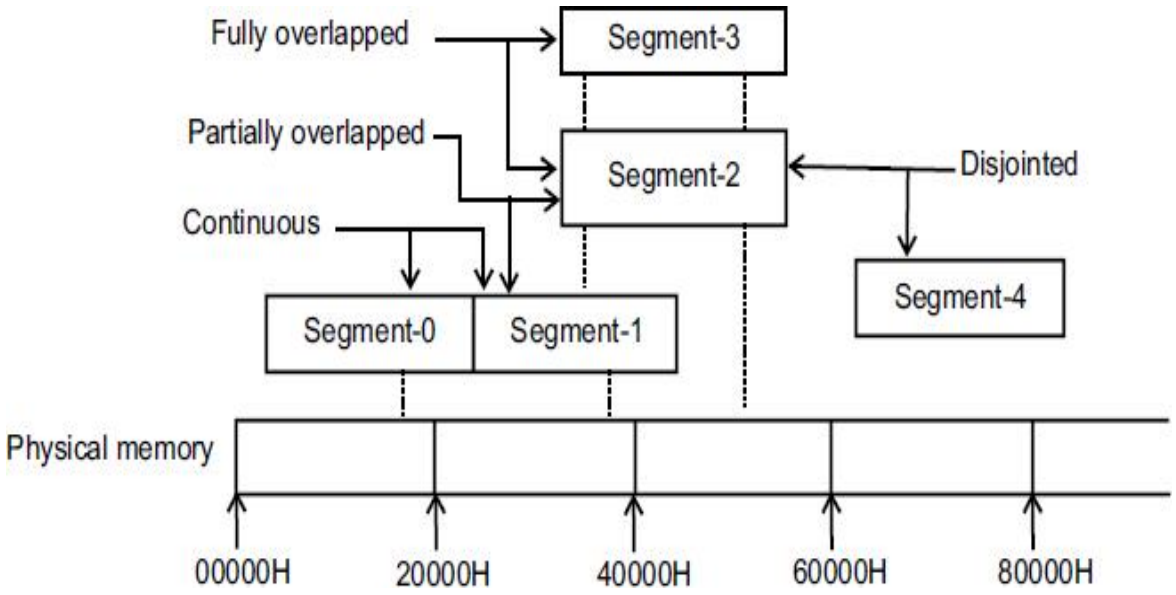
x86 Memory Partitioned into Segments

- 8086: maximum size is 64K (16-bit index reg.)
- 8086: can have 4 active segments (**CS, SS, DS, ES**)
- 8086: 2-data; 1-code; 1-stack
- x386: maximum size is 4GB (32-bit index reg.)
- x386: can have 6 active segments (4-data; **FS, GS**)

Why have segmented memory?

Other microprocessors could only address 64K since they only had a single 16-bit MAR (or smaller). Segments allowed computers to be built that could use more than 64K memory (but not all at the same time).


Code
FFFF
Fh
• Segment Registers:
–Point to Base Address
Segment
Extra
CS E S
SS D S
Stack
Fragmentation
• Index Registers:
–Contain Offset Value
Data
• Notation (Segmented Address):



Note that segments can overlap. This means that two different logical addresses can refer to the same physical address (aliasing).

In non-overlapping method, address of source is totally different from address of destination. Therefore, we can directly transfer the data using MOVSB/MOVSW instruction for transferring the data.

The blocks are said to be overlapped if some of the memory locations are common for both the blocks.

Case I: If address in SI is greater than address in DI then start the data transfer from last memory location keeping DF=1.

Case II: If address in SI is less than address in DI, then start the data transfer from first memory location by keeping DF=0.

2.3.4 Algorithm:

2.3.4.1 Non-overlapped mode

1. Initialize 2 memory blocks pointed by source and destination registers.
2. Initialize counter.
3. Move the contents pointed by source register to a register.
4. Increment address of source register.
5. Move the contents from register into location pointed by destination register.
6. Increment destination registers.
7. Decrement counter.
8. Repeat from steps 3 to step 6 until counter is 0.

9. End.

2.3.4.2 Non-overlapped moed

1. Initialize 2 memory blockspointed by source and destination registers.
2. Initialize counter.
3. Move the contents pointedby source register [si+count] to a variable.
4. Decrement address of sourecregister.
5. Move the contents from vairable into location pointed by destination register [di+count]
6. Decrement destination registers.
7. Decrement counter.
8. Repeat from steps 3 to step 6 until counter is 0.
9. End.

Input:

Array of number stored in location pointed by source and destination register Example:

Array 1 db 10h, 20h, 30h, 40h, 33h, 0ffh, 44,55h, 23h, 45h

Array2 db 00h, 00h, 00h, 00h, 00h, 00h, 00,00h, 00h, 00h

Output:

NON OVER LAPPED BLOCK TRANSFER

Array2 db 10h, 20h, 30h, 40h, 33h, 0ffh, 44,55h, 23h, 45h OVER

LAPPED BLOCK TRANSFER

Array2 db 00h, 00h, 00h, 00h, 00h,10h, 20h, 30h, 40h, 33h, 0ffh, 44,55h, 23h, 45h

Assignment Questions:

1. Memory: Even and odd banks
2. Address decoding techniques.
3. Comparison between memory mapped I/O and I/O mapped I/O.
4. Diagrammatic representation of the overlapped and non-overlapped block transfer
5. Comparison of overlapped and non-overlapped block transfer

Oral Questions:

1. Specify all the memory addressing instruction
2. What is the use of the Direction flag in Block transfer?
3. What is use of Source and Destination Index in above program
4. What is the change in the contents of memory locations in overlapped and non- overlapped mode?
5. Which interrupt is used to terminate the program in 8086 kit?

MPL Practical Oral Question Bank

Sr No	BL	Questions	Oral 1	Oral 2 (improvement)	Remark
1	1	Specify all the memory addressing instruction			
2	1	What is the use of the Direction flag in Block transfer?			
3	1	What is use of Source and Destination Index in above program			
4	1	What is the change in the contents of memory locations in overlapped and non- overlapped mode?			
5	1	Which interrupt is used to terminate the program in 8086 kit?			

Sign of Student