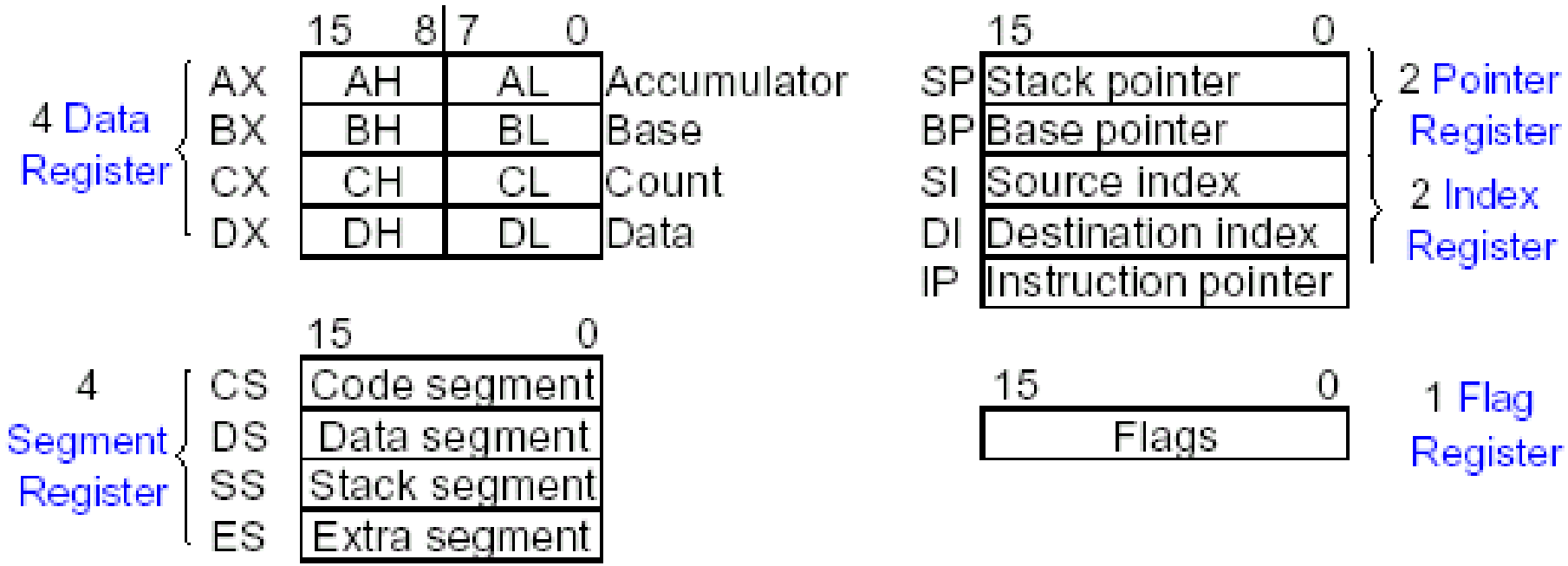
Microprocessor Programming Model

- The programming model (software) summarizes all the information needed for programming the microprocessor.
- Example: 8088 Programming Model



Program Visible and Invisible Registers

- The programming model of a microprocessor contains a concise description of the internal registers and their functions.
- Only the program visible registers (i.e. directly accessible by applications) are included in the programming model.
- All the registers in the Intel 8088 are program visible.

Intel 80X86 Registers

- Register in the 8088/8086 and 80286 may be grouped into 4 sets:

  (i) General registers (AX, BX, CX, DX)

  (ii) Pointer and index registers (SP, BP, SI, DI, IP)

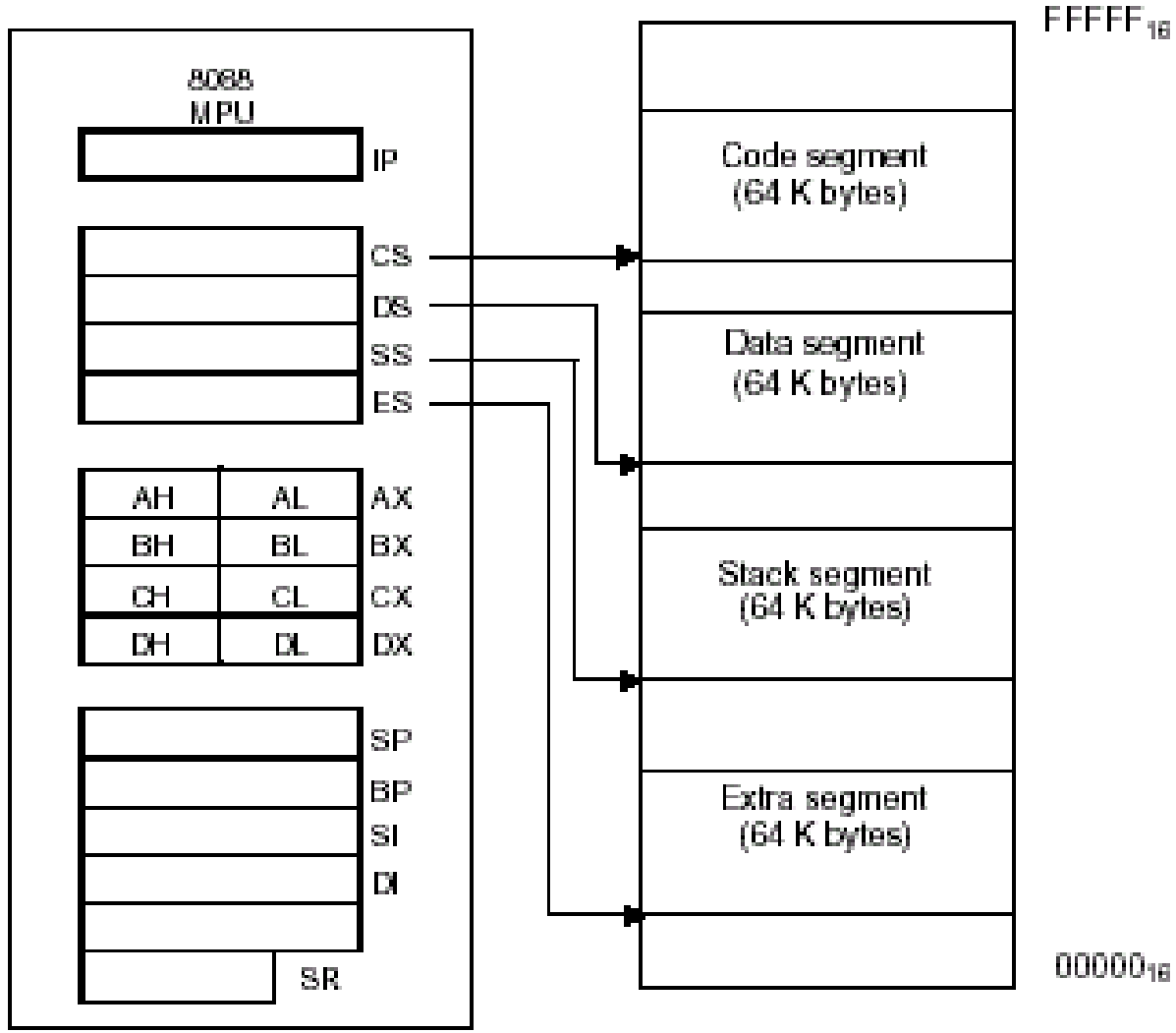
  (iii) Flag Register (FLAGS)

  (iv) Segment Registers (CS, DS, SS, ES)

- The 80386, 80486 and Pentium microprocessors have 32-bit registers and the 16-bit registers of the 8088 form a subset:

Software model of the 8088 microprocessor

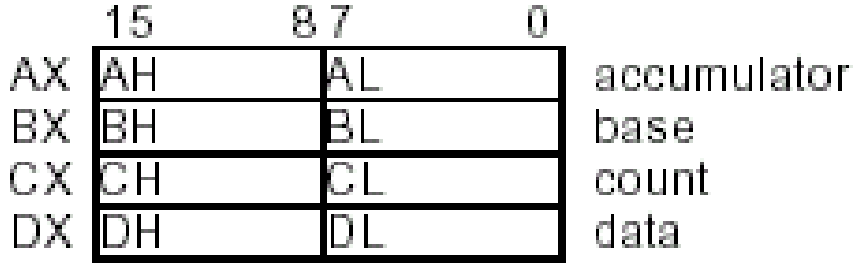
The segment registers in MPU store the initial address information of the corresponding memory segments.



SR: status (flag) register

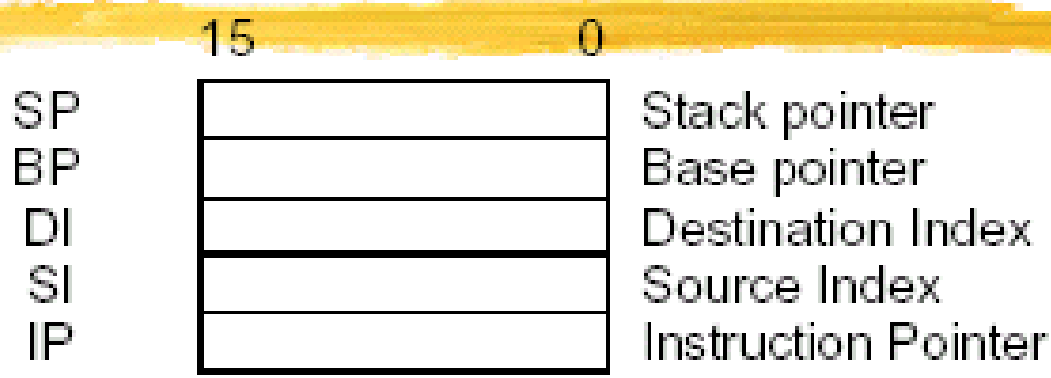
General Data Registers (AX, BX, CX, DX)

- AX, BX, CX, DX are 16-bit registers.
- The 16-bit registers in this set may be split into two.



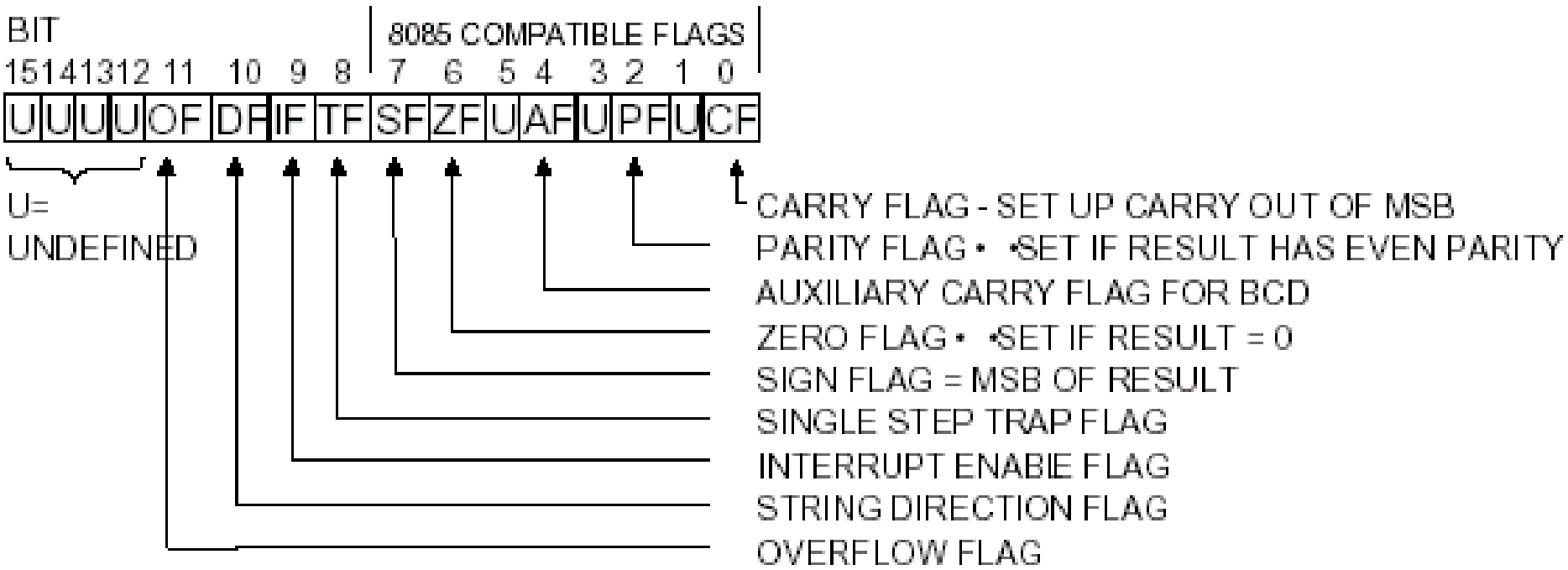
- AX (accumulator) stores the result of many arithmetic and logic instructions.
- BX (Base) stores the base (offset) address data in memory and the base address of a table of data referenced by the translate (XLAT) instruction.
- CX stores the count for certain instructions (eg. Counter in the LOOP instruction, the shift count for shift instructions).
- DX holds the most significant part of the result of a 16-bit multiplication, the most significant part of a dividend before division and I/O port number for a variable I/O instruction.

Pointer and Index Registers



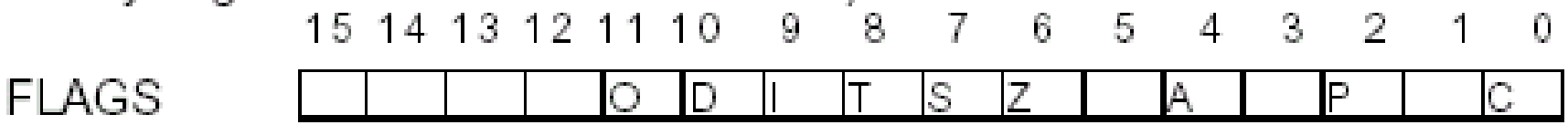
- This set of registers usually store offset addresses of memory. IP usually stores the offset address of the next instruction in memory, SP, BP, DI and SI usually store the offset address of data in memory.
- SP, BP, DI and SI may also be used for general purposes.
- SP (Stack Pointer) is used in the PUSH and POP instructions for operations on a LIFO (Last-In, First-Out) stack.
- BP(Base Pointer) is often used in addressing an array of data in the stack memory.
- DI (Destination Index) usually stores the indirect destination address of data from an instruction.
- SI (Source Index) is used when indirectly addressing source data in certain string instructions.

8086 Flag Register Format



Flags Register

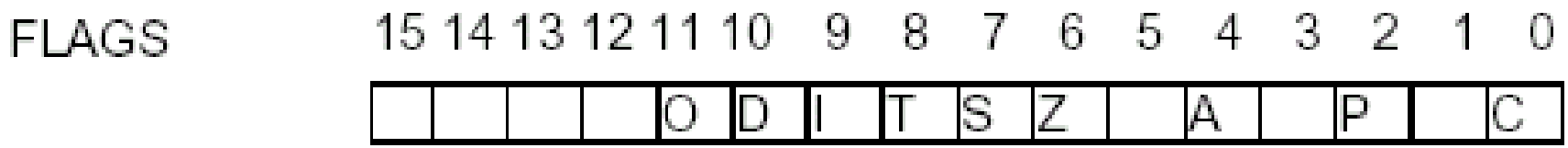
- 9 of the 16 bits in FLAGS can be set to one when certain events occur (ie they flag the occurrence of an event.) The other 7 bits are unused.



- All bits are set to zero on power up.
- Conditional jump instructions test the O, S, Z, P and C flags.
- Instructions (pushF, popF, LAHF, SAHF) are available for transferring the contents of the FLAGS register to/from the stack or to/from the AH register. Other instructions are available for manipulating certain flag bits (eg STI, CLI, STD, CLD, STC, CLC, CMC).

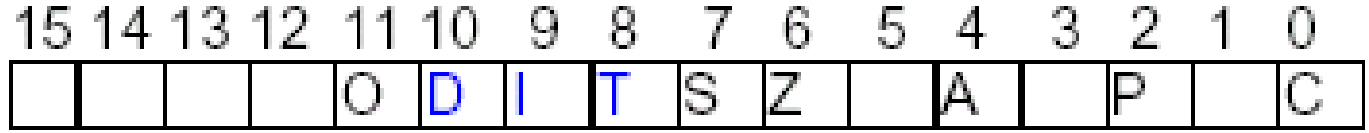
  eg. STI sets flag I to1, CLI sets flag I to 0, CMC complements flag C.

FLAGS



- Flags in the FLAGS register are either condition or control flags.
- Condition flag consist of :
  - C (carry flag) - set to 1 when the result of an addition has a carry out of the most significant byte. Other instructions can also affect C (eg subtraction)
  - P (parity flag) - set to 1 if the low order byte of the result contains an even number of ones; otherwise it is set to zero.
  - A (auxiliary carry flag) - set to one if there is a carry out of bit 3 during an addition or a borrow by bit 3 during subtraction.
  - Z (zero flag) - set to 1 if the result is zero; Z is otherwise zero.
  - S (sign flag) - equal to the most significant bit of the result (ie set to 1 if the result is negative)
  - O (overflow flag) - set if a result is out of range (eg when adding two positive numbers and the result appears negative)

Control Flags



FLAGS

- Three bits (D, I, T) in the flags register control the operation of the microprocessor under the following circumstances:
  - D (direction flag) - in certain string manipulation instructions, D determines whether the string is processed from the lowest address (D=0) or the highest address (D=1).
    D=0: auto-increment
    D=1: auto-decrement
  - I (interrupt flag) - determines whether a maskable interrupt is recognized by the microprocessor. If I=1, a maskable interrupt is possible, otherwise the interrupt is ignored.
  - T (trap flag) - if T=1, a trap (eg for single stepping through a program) is executed after every instruction.

Example of effect of an addition on the FLAGS register

If the following addition is carried out,

```
  0010 0011 0100 0101
+ 0011 0010 0001 1001
-------------------
  0101 0101 0101 1110
```

the FLAGS register's condition flags are set as follows:

S (sign) =0, Z (zero) =0, P (parity) =0, C (carry) =0,

A (aux carry) =0, O (overflow) =0

If instead the following were performed,

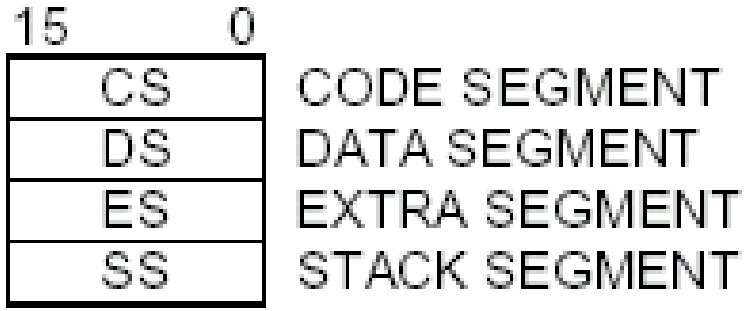
```
  0101 0100 0011 1001
+ 0100 0101 0110 1010
-------------------
  1001 1001 1010 0011
```

the FLAGS register's condition flags world be set as follows:

S (sign) =1, Z (zero) =0, P (parity) =1, C (carry) =0,

A (aux carry) =1, O (overflow) =1

Segment Registers



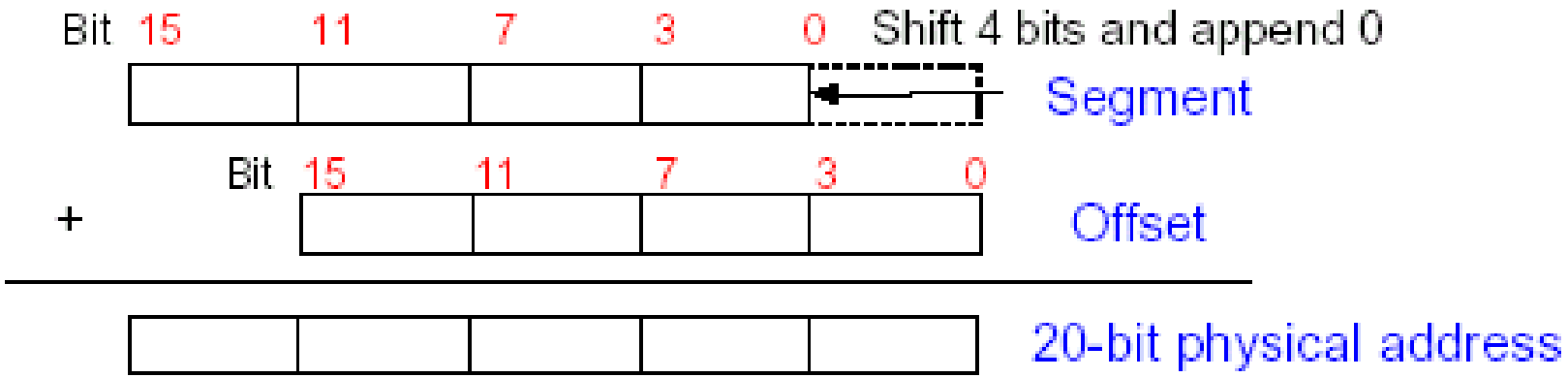
- Segment registers are 16-bit registers used in conjunction with the index and pointer registers to generate the physical 20-bit address.
- Code segment is the section of memory used to store the program instructions and procedures. CS stores the starting address of the program code. In the 8086 (and 80286) the code segment is limited to 64Kbytes in length (in 80386 the maximum length is 4Gbytes).
- Data segment (DS) contains data used by the program. DS stores the starting address of the data segment.
- Extra segment is another data segment which is used by some string instructions.
- Stack segment (SS) is the section in memory called stack which is used for storage of register contents and data.

Segmentation

- A segment in the 8088 system is a continuous section of memory of up to 64Kbytes in length.
- Since the segment address is shifted by 4 bits to form the 20-bit address, the start address of a segment can only occur at 16 data byte intervals. Addresses at every 16 data byte interval are valid for starting segments. (*Increasing the segment value by 1 will increase the physical starting address by 16.*)
- The 64-Kbyte block defined by the start address of a segment may overlap with other segments or be completely in its own separate area of memory.
- Segments allow packages of information (eg a data table, or a subroutine) to be kept separate - it is not necessary to fill all 64K of the segment and the programmer can make the segment of arbitrary size (up to 64K, in 16 byte increments).

Segments and Offsets

- The 20-bit address in memory is generated by adding a 16-bit offset to a 16-bit segment address. The segment address defines the start of a 64kbyte memory block within the 1Mbyte address space, and the offset defines the exact memory location within that 64Kbyte block.
- A 20-bit address is formed by shifting the 16-bit segment address by 4 bits and adding to the 16-bit offset.



Segments and Offsets (Cont.)

- Advantages of the segment + offset method include

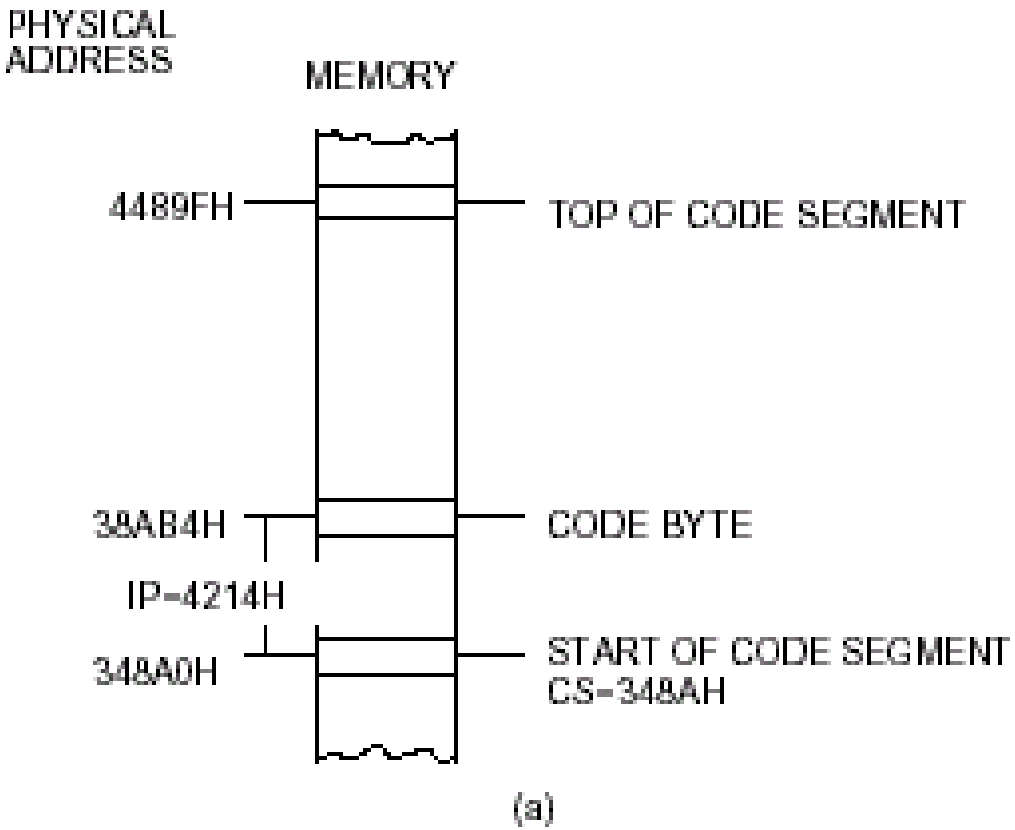
1. Program code can easily be reallocated in memory (useful for multi-tasking)
2. Most operations can be performed by changing only the 16 bit offset. The offset can be stored in 16-bit registers (20-bit registers are not necessary), allowing for easier interface to 8- and 16-bit wide memory.

Rules of combination of Segment registers and Offset

- The microprocessor has a set of rules that define the segment register and offset register combination used by certain addressing modes.
- However, the default can be overrode by using the segment override prefix e.g. MOV CL, [BP] or MOV CL,DS:[BP]

Offset register	Default Segment register	Override Prefix
IP	CS	Never
SP	SS	Never
BP	SS	DS, ES or CS
SI, DI (not include strings)	DS	ES, SS or CS
DI for string instructions	ES	Never

Addition of IP to CS to produce the physical address of the code type



(a)

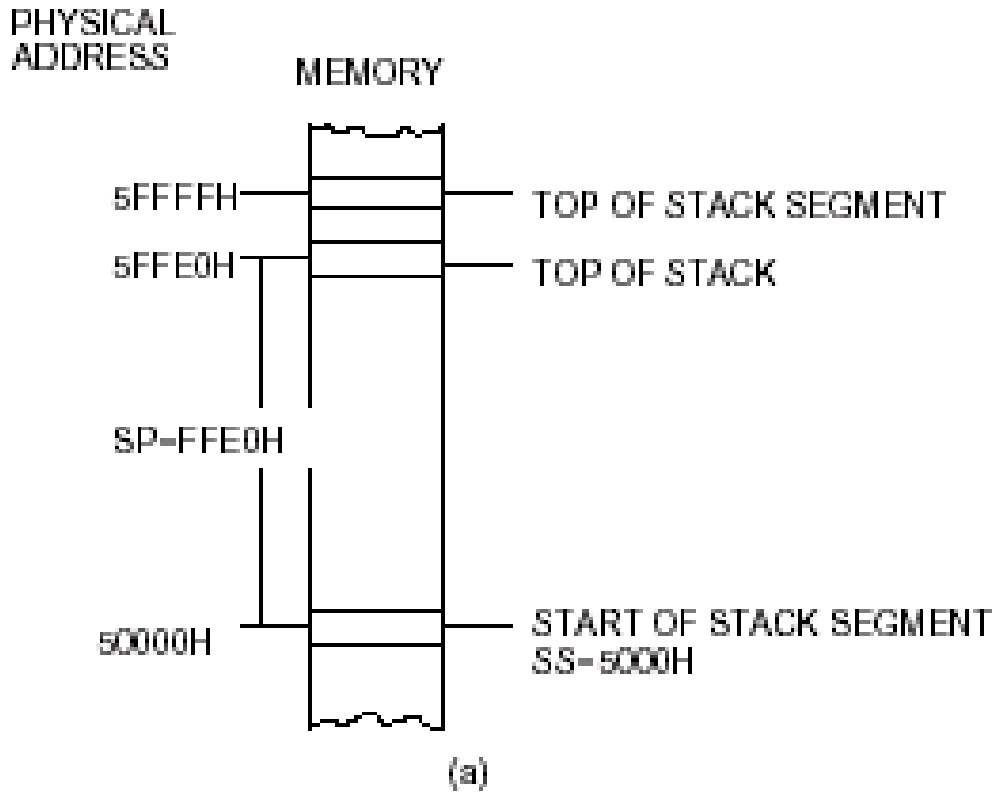
Diagram

CS	3	4	8	A	0	— HARDWIRED ZERO
IP +		4	2	1	4	
PHYSICAL ADDRESS	3	8	A	B	4	PHYSICAL ADDRESS

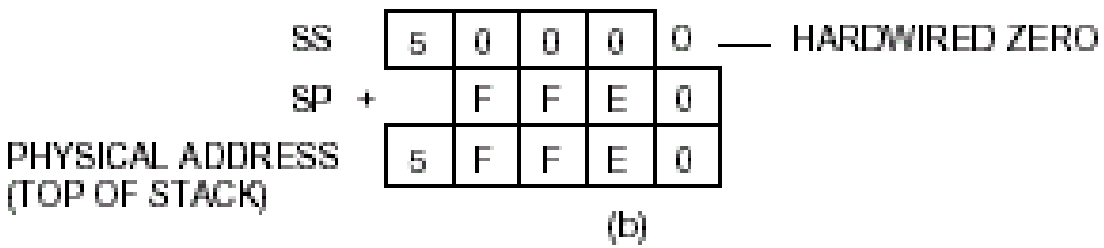
(b)

Computation

Addition of SS and SP to produce the physical address of the top of the stack

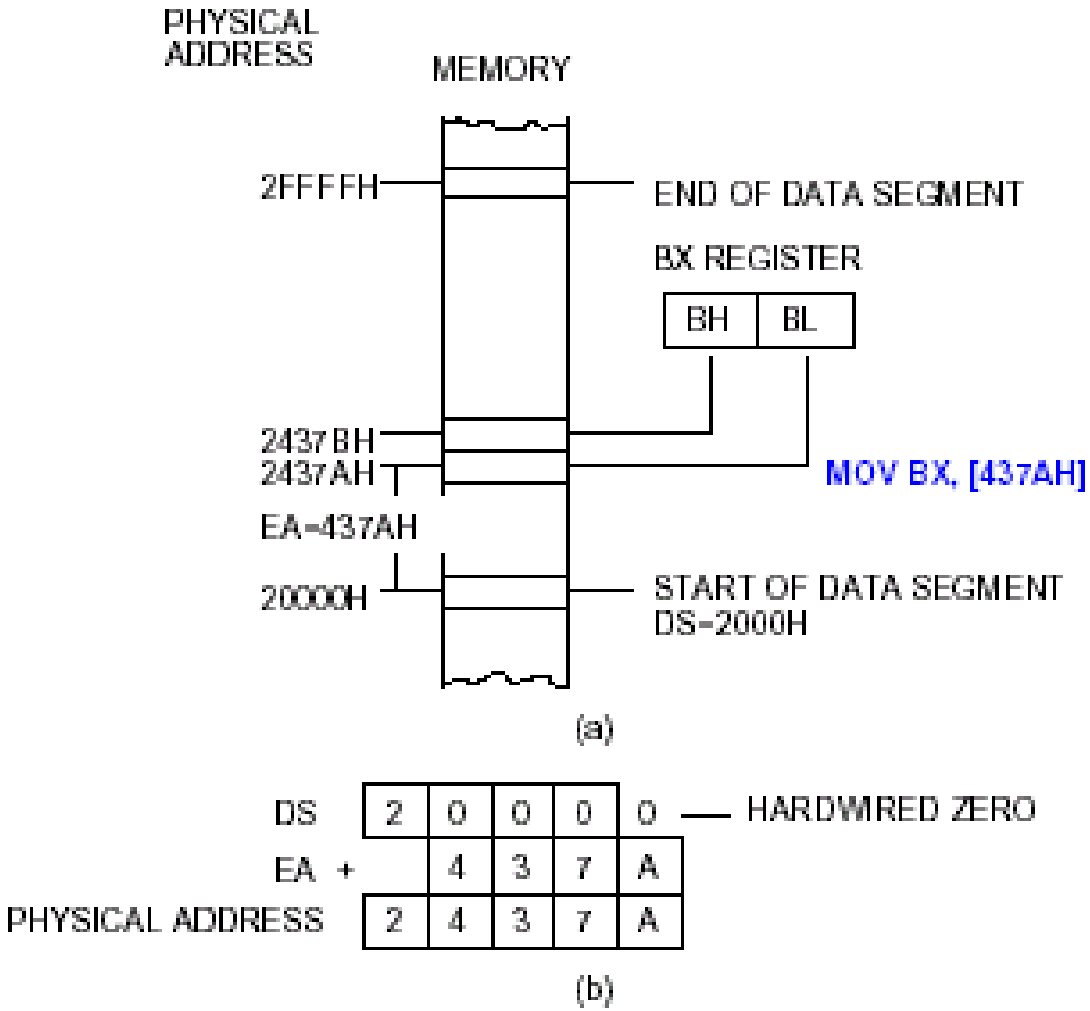


Diagram



Computation

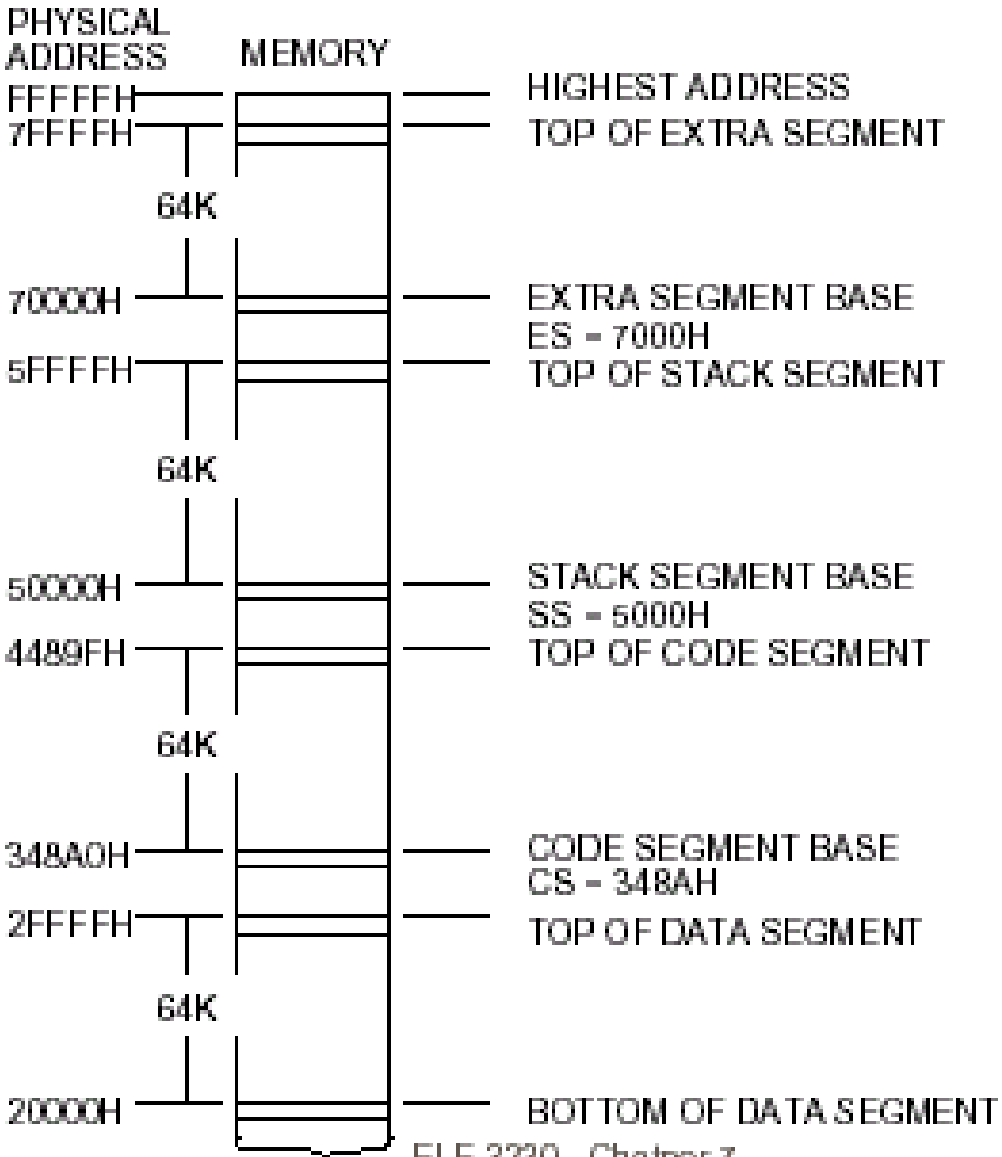
Addition of data segment register and effective address to produce the physical address of the data byte



Diagram

Computation

One way four 64-Kbyte segments might be positioned within the 1-Mbyte address space of an 8086

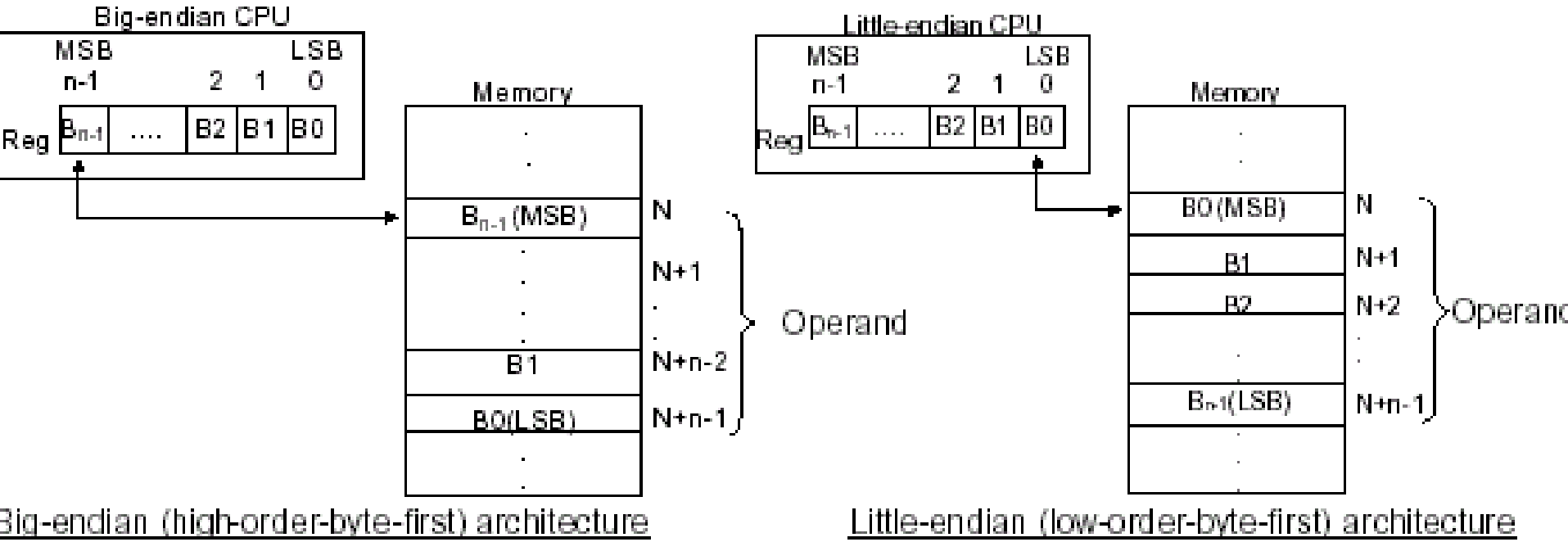


Number in specific microprocessors

Number with several bytes may be stored in 2 ways:

- Low Order Byte First (LOBF) architecture (also called little endian) - the least significant byte is put in the first (lowest) memory address. Examples of microprocessors which use this convention include Intel 8086/8088
- High Order Byte First (HOBF) architecture (also called big endian) - the most significant byte is put first. Example include the Motorola 680X0 microprocessors

Number in specific microprocessors



Big-endian (high-order-byte-first) architecture

Little-endian (low-order-byte-first) architecture

FAQ

- The overflow flag will be set if the resultant number is too big (eg when adding two positive number) or too small (eg when adding two negative numbers) to present.

Example

```
mov al,0a0h
mov ah,0b0h
add al,ah
```

The result is ax=0b050h with C=1, Z=0, S=0, O=1, P=1, A=0.