An Introduction to the 8088 Microprocessor (1)

Presentation · November 2013

CITATIONS
0

READS
13,225

1 author:

Abdullatif Baba
Kuwait College of Science and Technology (Private University)
109 PUBLICATIONS **98** CITATIONS

SEE PROFILE

An Introduction to the 8088 Microprocessor

Dr. Eng. Abdellatif BABA

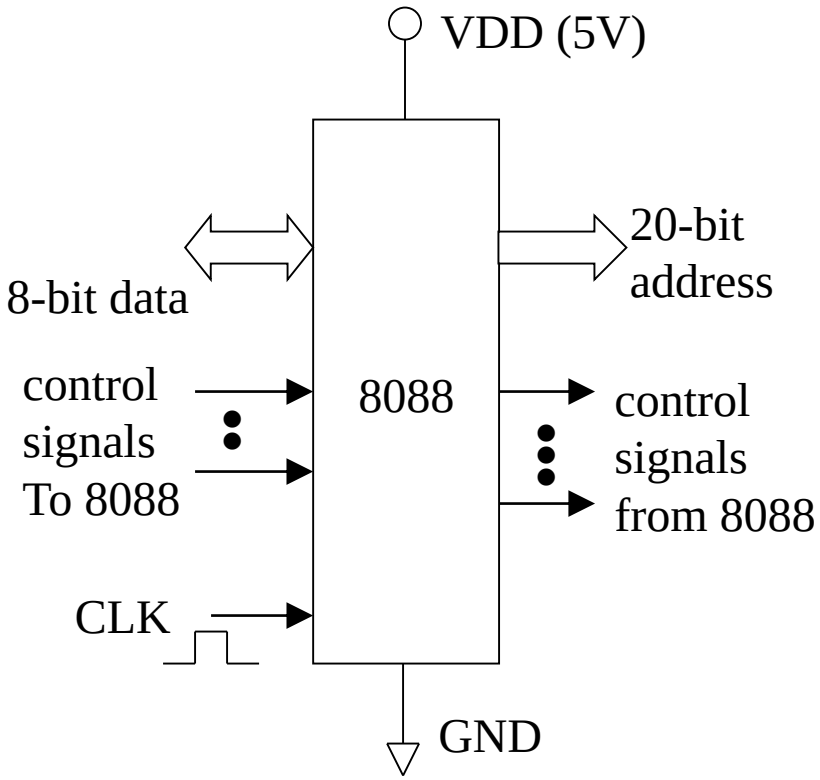
Overview

- Intel 8088 facts

- 20 bit address bus allow accessing 1 M memory locations

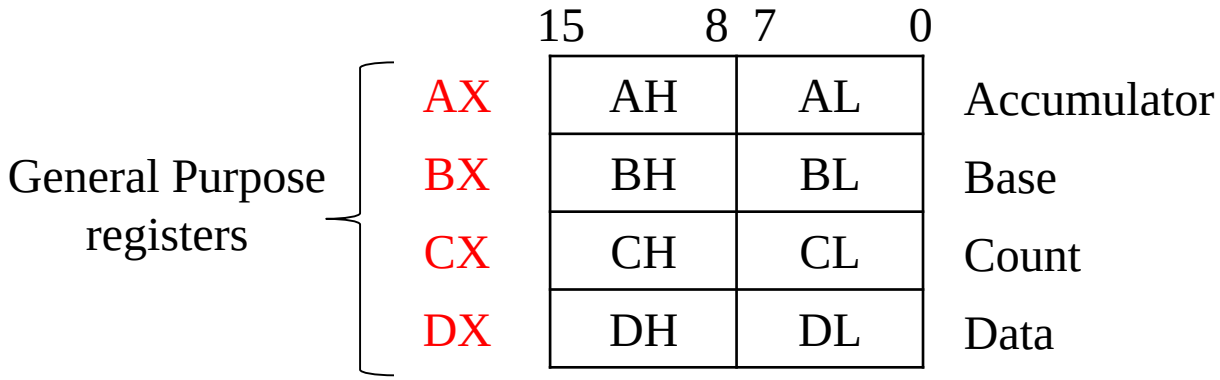
- 16-bit internal data bus and 8-bit external data bus. Thus, it need two read (or write) operations to read (or write) a 16-bit datum

20 address lines => 2^{20} = 1MB (Mega Byte)

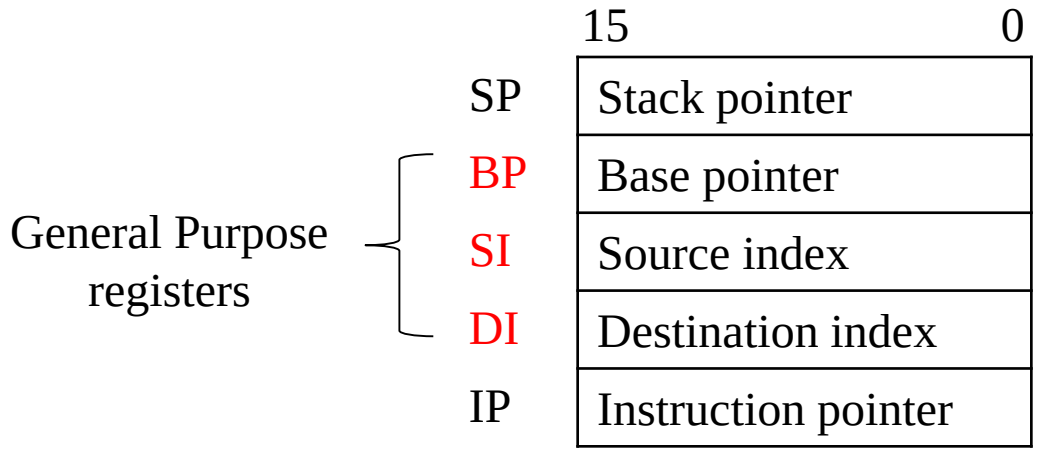


8088 signal classification

The Software model of the 8088

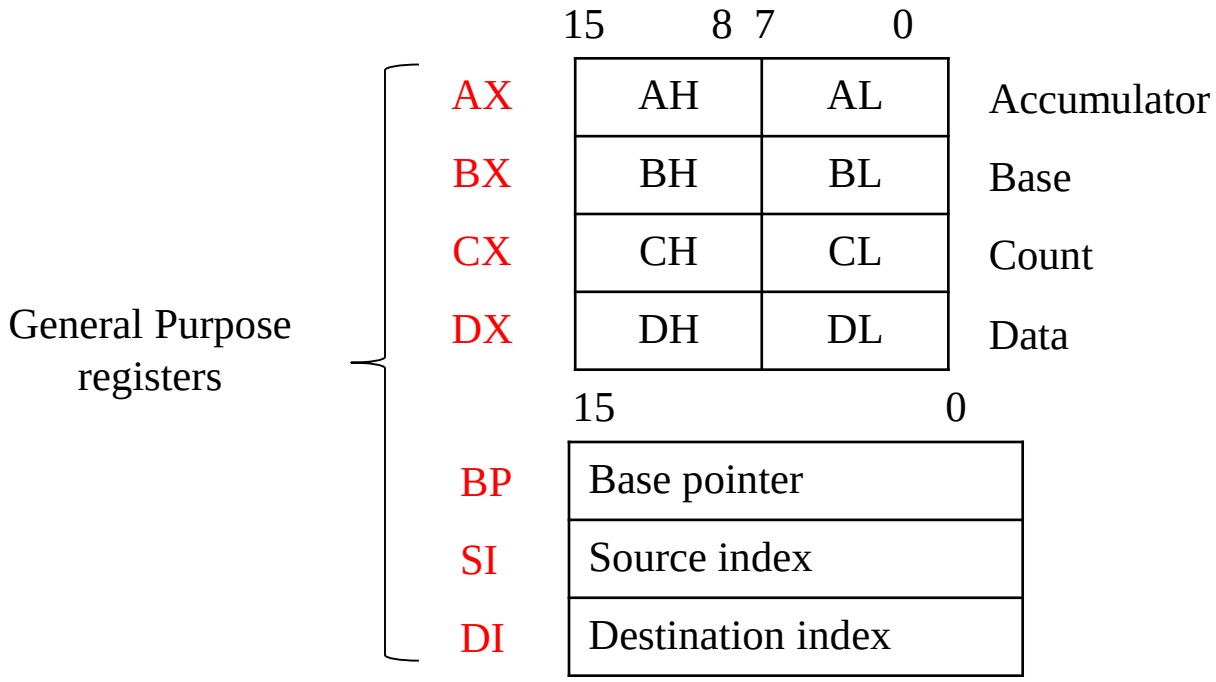


Four 16-bit data registers, each one may be split up into two halves of 8 bits



Five 16-bit registers are available to use as pointer or index registers
Non of them may be divided up

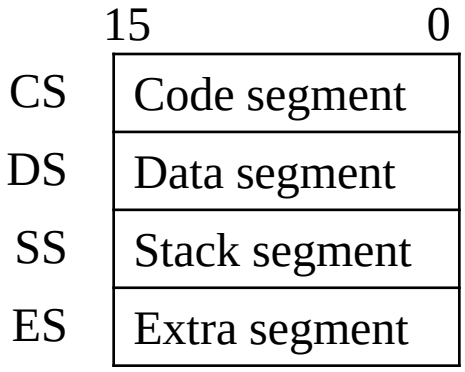
The Software model of the 8088



7 general purpose 16-bit registers are available to be used by the programmer, they have some specific roles :

- AX : is normally used in multiplication and division operations and also in instructions that access I/O ports.
- CX : is normally used as a counter in loop operations providing up to 65,536 counts
- DX : is may be used in multiplication and division operations and also a pointer when accessing I/O ports
- SI and DI : are used as pointers in string operations.

The Software model of the 8088



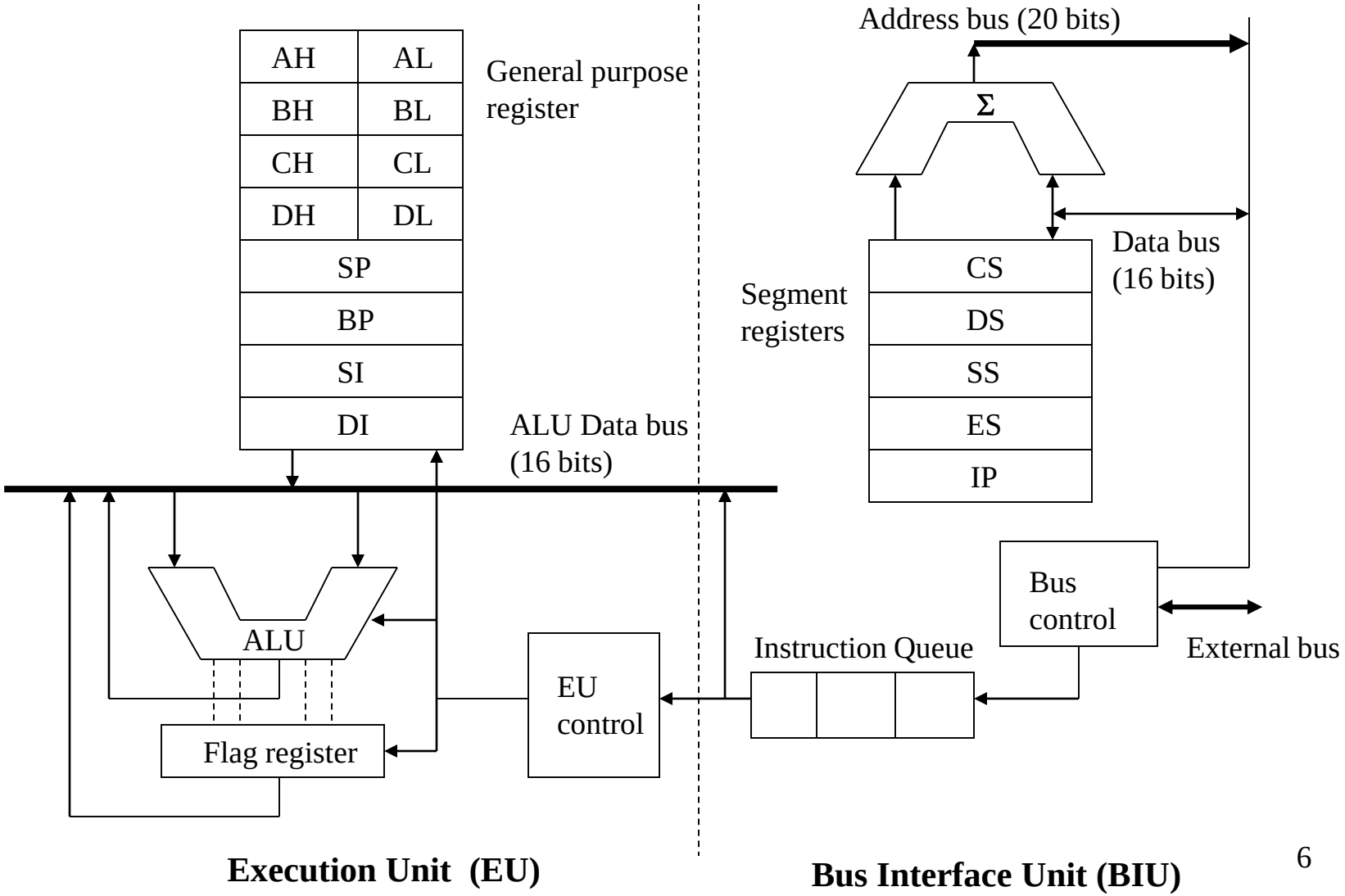
Segment registers, are used by the processor to control all access to memory and I/O and must be maintained by the programmer

- CS: used during instruction fetch
- DS: used by default when reading or writing data
- SS : used during stack operations such as subroutine calls and returns
- ES : used fore anything the programmer wish

To indicate the result of arithmetic and logical instructions. (Zero, parity, sign, carry…..)

8088 is composed of two internally divided functional units

- Bus interface unit (BIU)
- Execution unit (EU)



Organization of 8088

Bus Interface unit (BIU) :

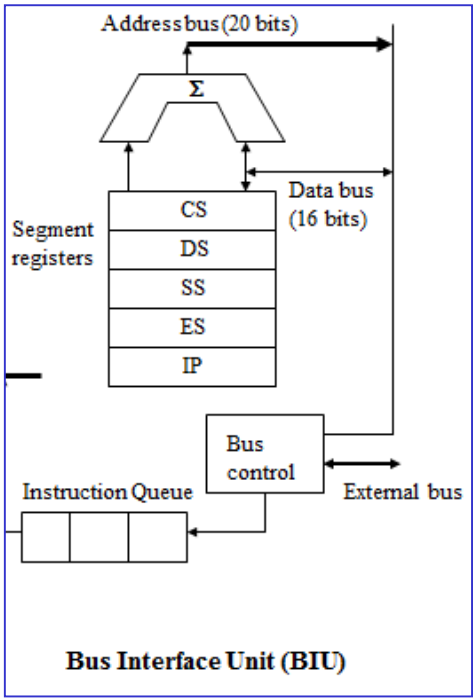
BIU is responsible for performing all memory and I/O accesses.

BIU is composed of :

- Address adder
- Segment registers
- Bus control unit
- The instructions queue.

In any microprocessor the following cycle has to be achieved

Fetch an instruction + decode it + execute it



Bus Interface Unit (BIU)

Organization of 8088

Bus Interface unit (BIU) :

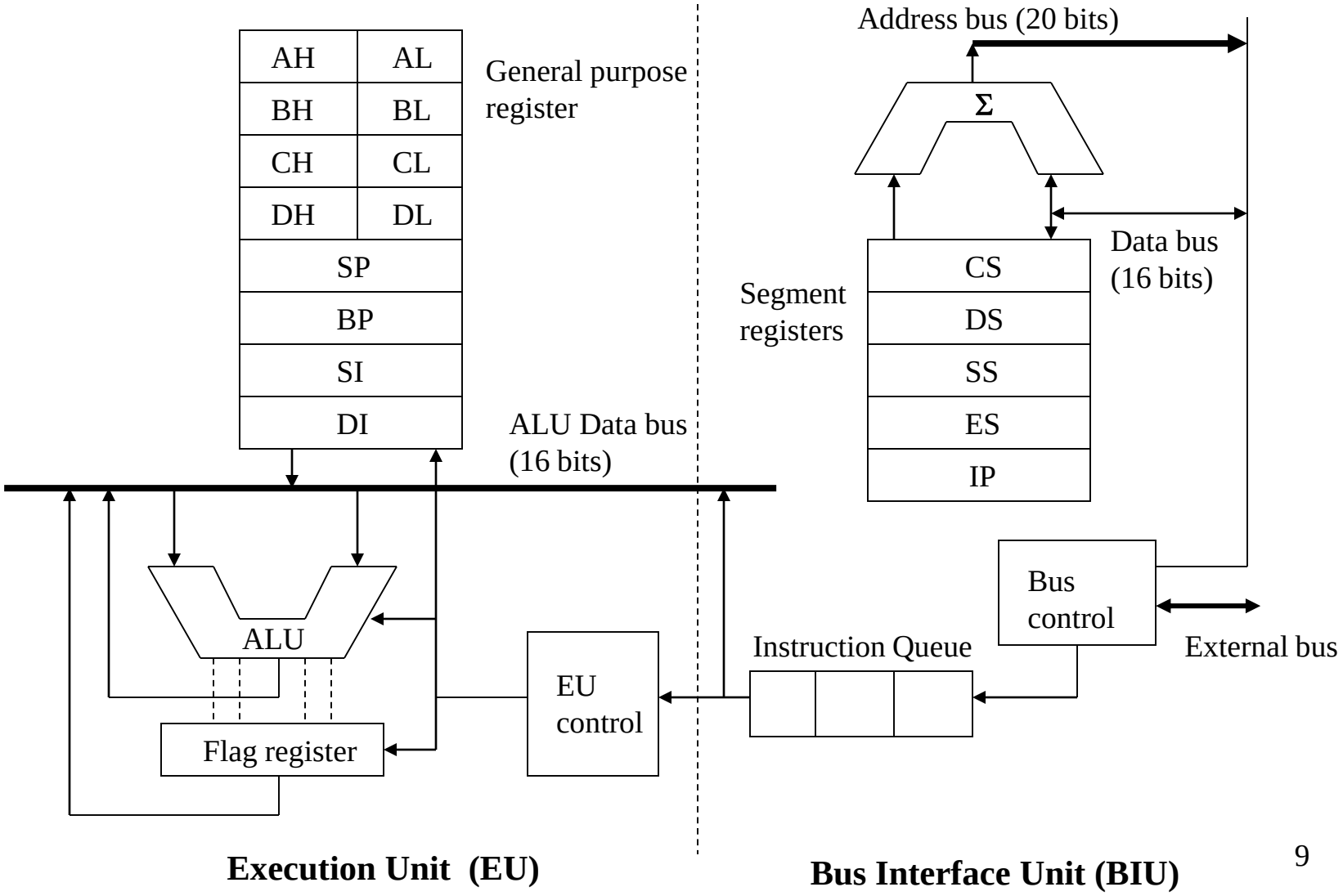
- During an instruction fetch, BIU passes the instruction to the EU. And then pre-fetch more instructions to be stored in the Instruction queue (*Which is an internal FIFO queue designed to hold 4 bytes of code in memory following the current instruction*).
- When the EU is ready for the next byte, it will be pulled from the queue.
- BIU pre-fetch the next byte from the memory and so on …..

When the current instruction is Jump, subroutine call or Return instruction.

- The instructions pre-fetched by the BIU are incorrect and are discarded
- The next (appropriate) instruction is fetched from the new address and the queue is reloaded

8088 is composed of two internally divided functional units

- Bus interface unit (BIU)
- Execution unit (EU)



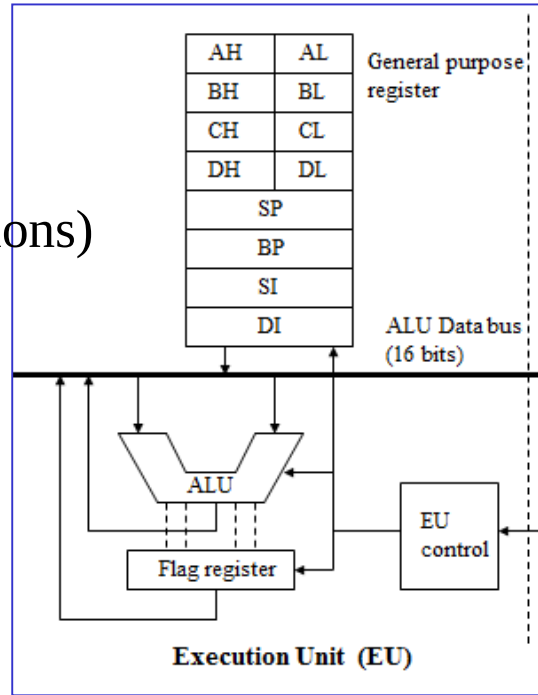
Organization of 8088

Execution unit (EU) :

- **Houses Arithmetic and logic unit**
- **Responsible for executing program instructions provided to it by 16 bit data bus.**
- **Maintain the Flag register**
- **All instructions and data arrive the EU from BIU**

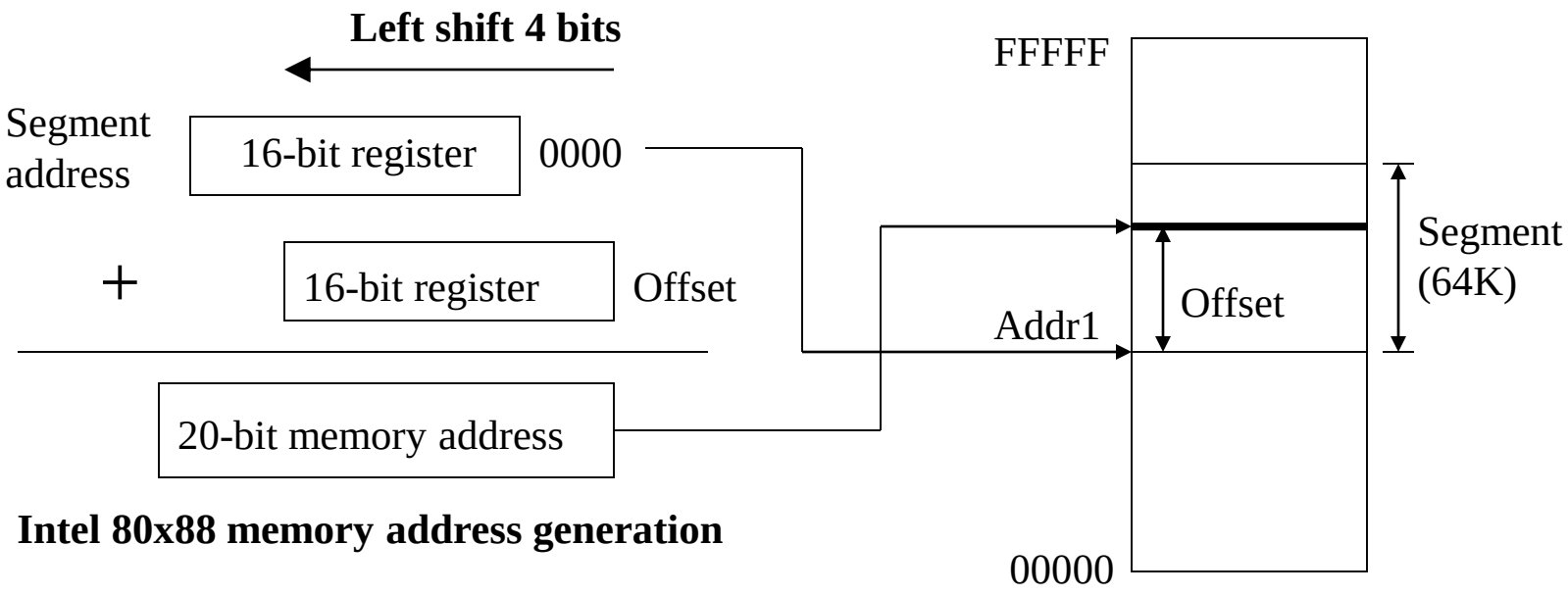
EU is composed of :

- ALU (Handling Arithmetic and logic operations)
- All general purpose registers
- Pointer registers
- The execution control unit.



Generating Memory Addresses

- How can a 16-bit microprocessor generate 20-bit memory addresses?



Intel 80x88 memory address generation

1M memory space

Remember

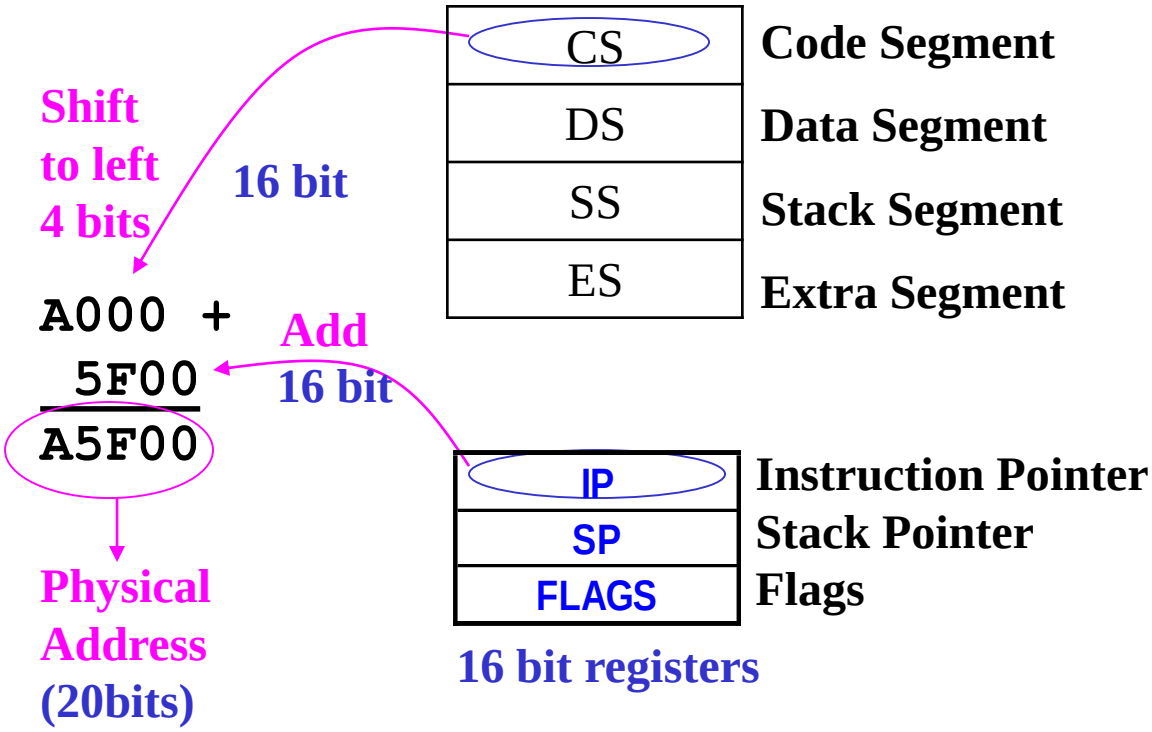
CS	**Code Segment**
DS	**Data Segment**
SS	**Stack Segment**
ES	**Extra Segment**

Segment = a 64kbyte memory block beginning at a multiple by 10H address.

Generating Memory Addresses

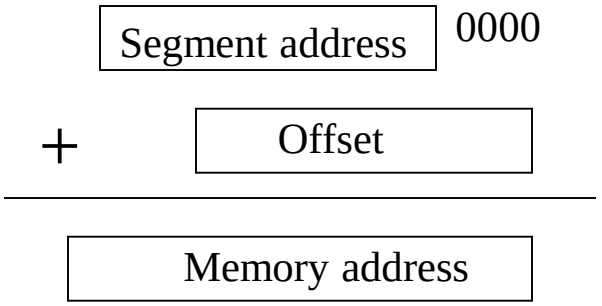
An physical address is generated as combination between a segment register and another register as in the following example.

Example :



Memory Address Calculation

- Segment addresses must be stored in segment registers
- Offset is derived from the combination of pointer registers, the Instruction Pointer (IP), or immediate values



- Examples

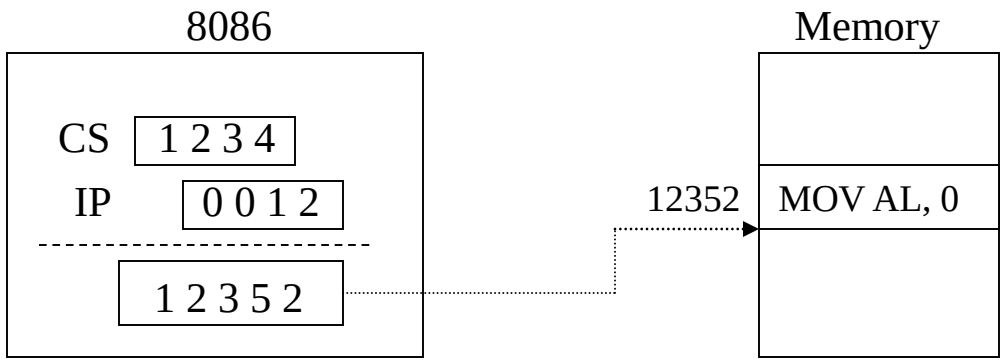
CS	3	4	8	A	0
IP +		4	2	1	4
Instruction address	3	8	A	B	4

SS	5	0	0	0	0
SP +		F	F	E	0
Stack address	5	F	F	E	0

DS	1	2	3	4	0
DI +		0	0	2	2
Data address	1	2	3	6	2

Fetching Instructions

- Where to fetch the next instruction?



- Update IP
  - — After an instruction is fetched, Register IP is updated as follows:

    IP = IP + Length of the fetched instruction

  - — For Example: the length of **MOV AL, 0** is 2 bytes. After fetching this instruction, the IP is updated to 0014

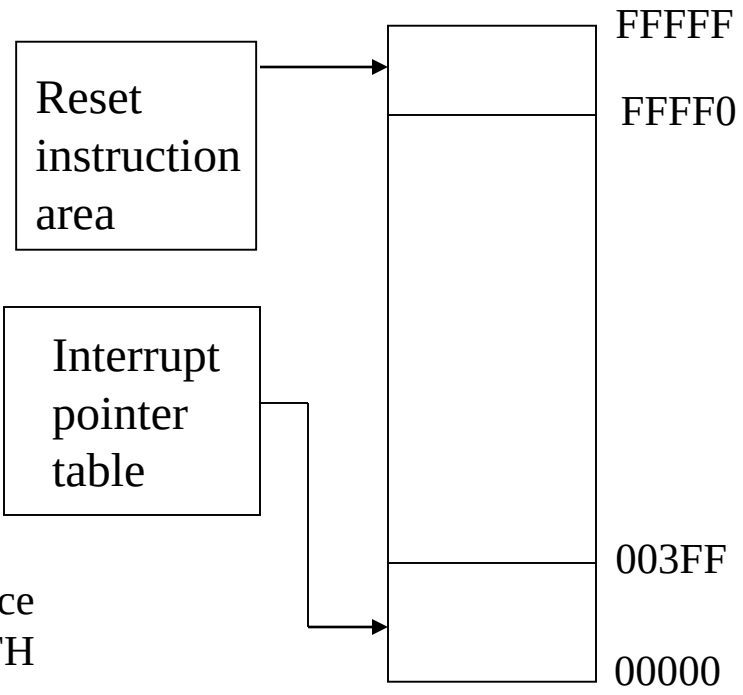
Reserved Memory Locations

- Some memory locations are reserved for special purposes. Programs should not be loaded in these areas

  - Locations from FFFF0H to FFFFFH are used for system reset code

  - Locations from 00000H to 003FFH are used for the interrupt pointer table
    - It has 256 table entries
    - Each table entry is 4 bytes

    256 × 4 = 1024 = memory addressing space
    From 00000H to 003FFH



Flag register

❑ Flag register contains information reflecting the current status of a microprocessor. It also contains information which controls the operation of the microprocessor.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-				OF	DF	IF	TF	SF	ZF	-	AF	-	PF	-	CF

CF	**Carry Flag**	**Contains Carry out of MSB of result**
PF	**Parity Flag**	**Indicates if result has even parity**
AF	**Auxiliary carry Flag**	**Contains Carry out of bit 3 in AL**
ZF	**Zero Flag**	**Indicates if result equals zero**
SF	**Sign Flag**	**Indicates if result is negative**
TF	**Trace Flag**	**Provides a single step capability for debugging**
IF	**Interrupt enable Flag**	**Enables/disables interrupts**
DF	**Direction Flag**	**Controls pointer updating during string operations**
OF	**Overflow Flag**	**Indicates that an overflow occurred in result**

DF, IF and TF are Control Flags, the others are called Status Flags

Carry Flag (CF): Bit 0

- The carry out or borrow out from MSB in the last arithmetical operation.

Bit 7 for byte operation
Bit 15 for word operation
Bit 31 for double-word operation

- CMP (Compare) instruction is a subtraction without a saved result. Carry bit is affected.

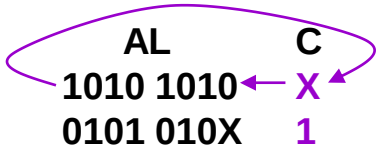
- CF is not affected by other types of instructions (i.e. MOV, JMP, etc)

Examples:

```
MOV  AL, FFH
INC  AL
```

Flag not affected (keeps the value corresponding to the previous arithmetical operation).

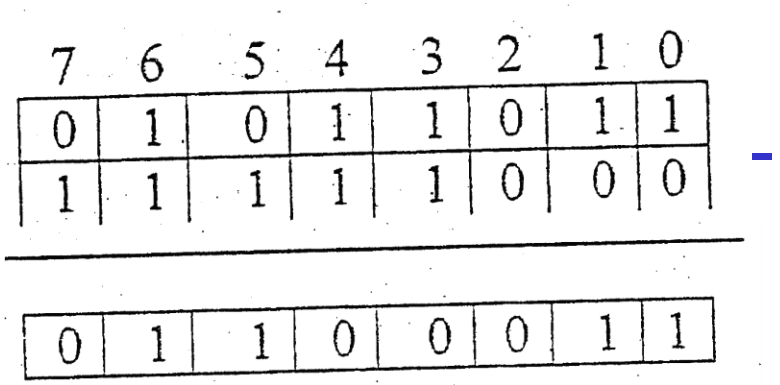
```
MOV  AL, AAH
RCL  AL
```

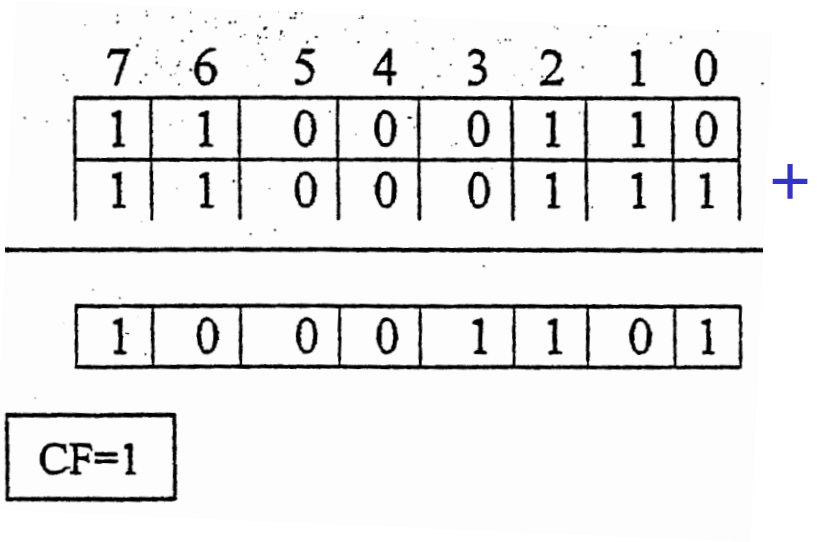


Rotate Left trough Carry

Carry Flag (CF): Bit 0

Examples :





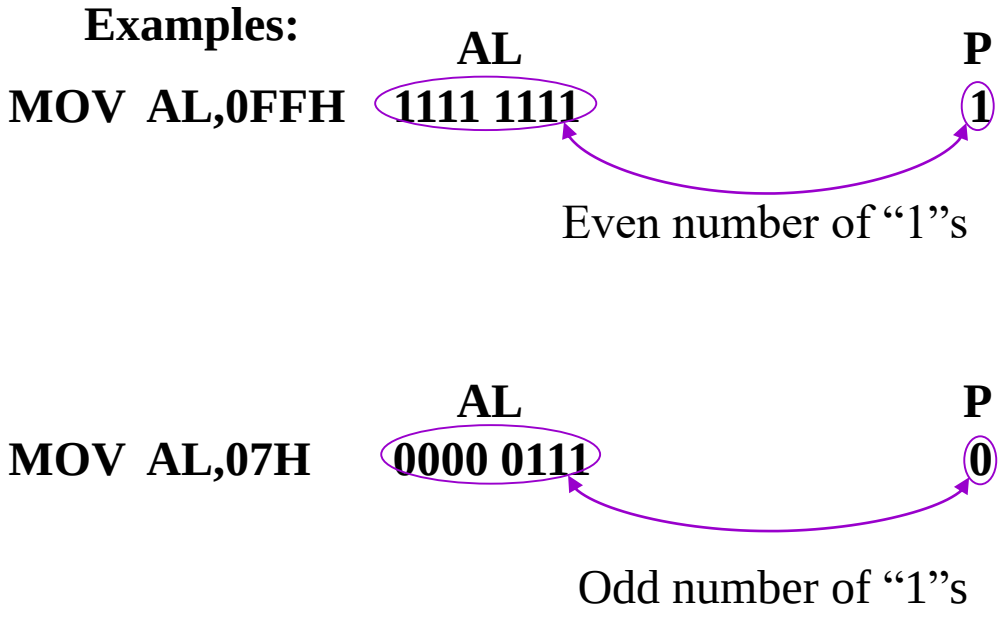
CF = 1; if there is a carry out or borrow out from MSB in the last arithmetical operation.

Otherwise CF = 0;

Parity Flag (PF): Bit 2

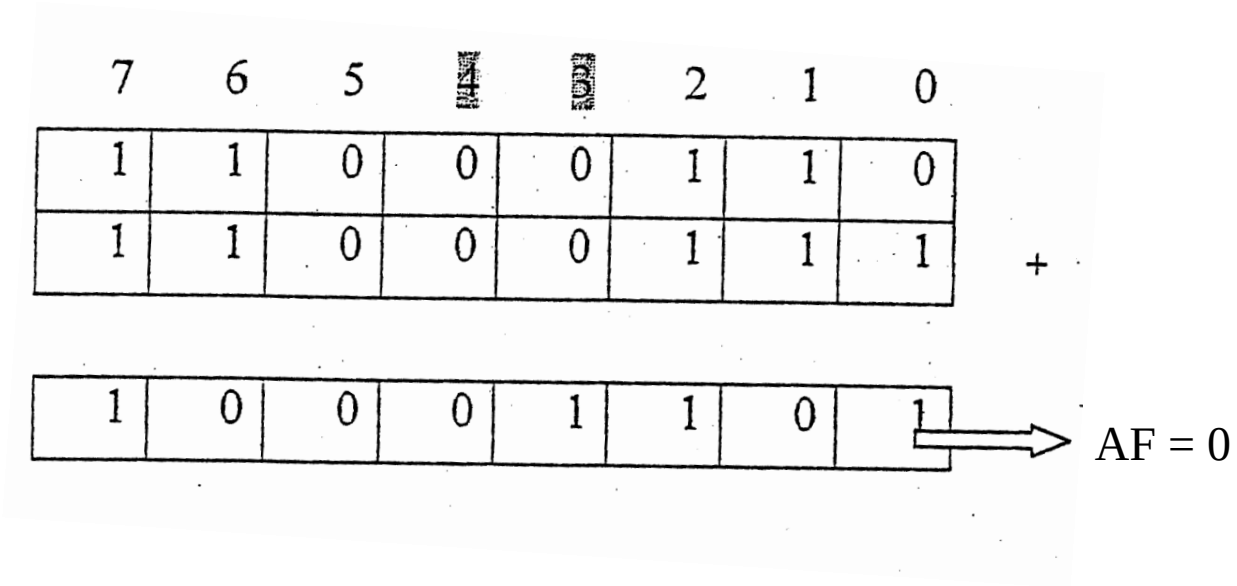
Set (1) if an even number of bits in the lower byte of the result are "1".

Examples:



Auxiliary Carry Flag (AF): Bit 4

The carry out from bit 3 to bit 4 in the last arithmetical operation.



AF = 1; if there is a carry out from 3rd bit to the 4th bit
Otherwise AF = 0;

Zero Flag (ZF): Bit 6

Set (1) if the result of the last arithmetical or logical operation was 0.

0000 0000 for byte operation

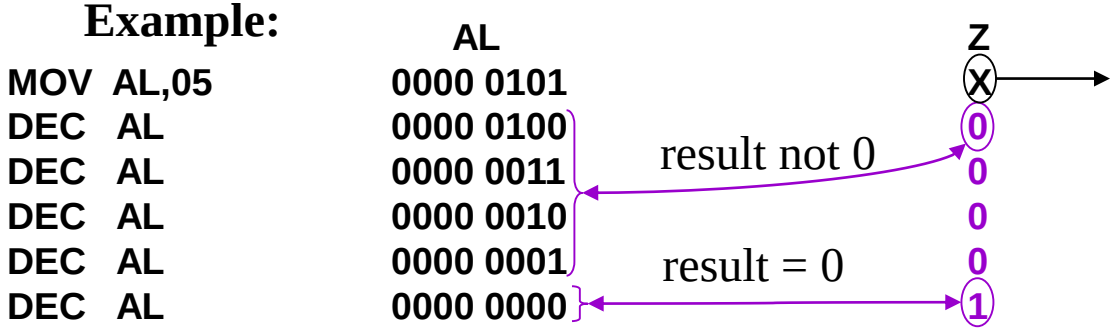
0000 0000 0000 0000 for word operation

0000 0000 0000 0000 0000 0000 0000 0000 for double-word operation

CMP (Compare) instruction is a subtraction without a saved result. Zero bit is affected.

Not affected by other types of instructions (i.e. MOV, JMP, etc)

Example:



Flag not affected (keeps the value corresponding to the previous arithmetical operation).

Sign Flag (SF): Bit 7

Represent the sign of the last arithmetical or logical operation.

The MSB (Most Significant Bit) of the last arithmetical or logical operation result.

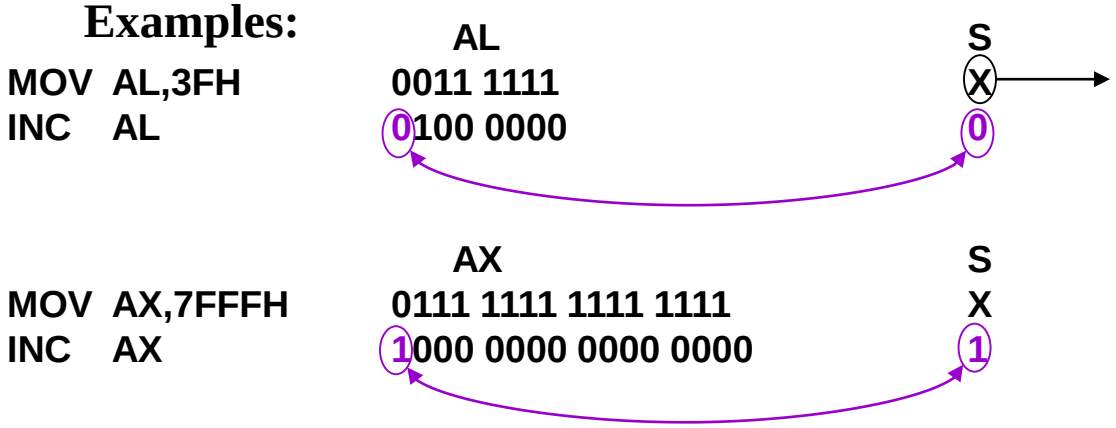
Bit 7 for byte operation
Bit 15 for word operation
Bit 31 for double-word operation

The processor doesn't knows if the result is to be interpreted as "signed" or "unsigned". S flag is always generated. It is the programmer responsibility to test or not the S flag.

CMP (Compare) instruction is a subtraction without a saved result. Sign bit is affected.

Not affected by other types of instructions (i.e. MOV, JMP, etc)

Examples:



Flag not affected (keeps the value corresponding to the previous arithmetical operation).

Overflow Flag (OF): Bit 11

It becomes 1 if the last calculated result would not fit in the number of bits used for the operation, this may happen in the following cases :

Example :

Signed numbers: +127+127 has to be +254 but in binary 1111 1110
Thus it negative result .. So it is an overflow case has to be detected

The overflow flag is set when the most significant bit (the sign bit) is changed by adding two numbers with the same sign (or subtracting two numbers with opposite signs).

Overflow never occurs when the sign of two addition operands are different (or the sign of two subtraction operands are the same).

Flag register

Interrupt enable Flag (IF): Bit 9

It is manipulated by the programmer. If the flag is set to 1, hardware interrupts will be authorised. If it is set to 0, interrupts will be ignored.

Trap Flag (TF): Bit 8

It is used for single step control. It allows user to execute one instruction of a program at a time for debugging. When trap flag is set, program can be run in single step mode.

Direction Flag (DF): Bit 10

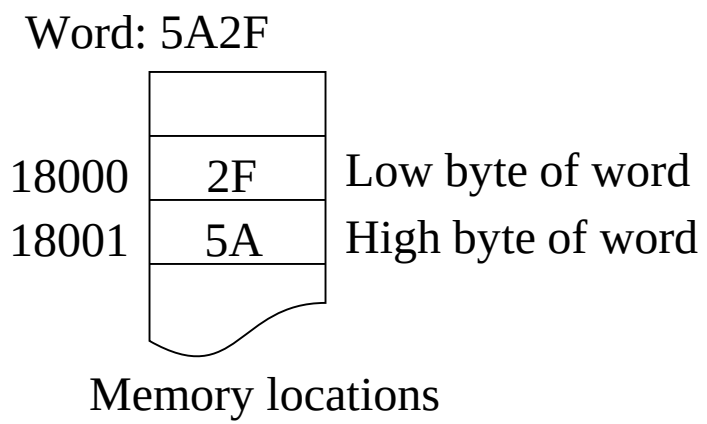
Is used for the direction of strings. If it is set, string bytes are accessed from higher memory address to lower memory address. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

Data Organization

Bits, Bytes, Words, Double-words

			Possible Values	
Name	**Size**	**Binary**	**Hexadecimal**	**Decimal**
Bit	**BInary digiT**	**0,1**	**0,1**	**0,1**
Nibble	**4 bits**	**0...1111**	**0...F**	**0…15**
Byte	**8 bits**	**0…1111,1111**	**0…FF**	**0…255**
Word	**16 bits = 2 bytes**	**0…(16 '1's)**	**0…FFFF**	**0…65,535**
Double Word	**32 bits = 4 bytes**	**0…(32 '1's)**	**0…FFFFFFFF**	**0...4,294,967,295**

Byte swapping: if a word has to be stored into an 8 bit wide memory at address adr, its low byte is stored at adr and its high byte at adr+1. If a word is read from an 8 bit memory at address adr, the low byte is loaded from adr and the high byte from adr+1.



Rule: ***low significance <=> low address***

Instruction types

Data transfer instructions

8088 instruction set

IN	Input byte or word from port
LAHF	Load AH from flags
LDS	Load pointer using data segment
LEA	Load effective address
LES	Load pointer using extra segment
MOV	Move to/from register/memory
OUT	Output byte or word to port
POP	Pop word off stack
POPF	Pop flags off stack
PUSH	Push word onto stack
PUSHF	Push flags onto stack
SAHF	Store AH into flags
XCHG	Exchange byte or word
XLAT	Translate byte

Additional 80286 instructions

INS	Input string from port
OUTS	Output string to port
POPA	Pop all registers
PUSHA	Push all registers

Additional 80386 instructions

LFS	Load pointer using FS
LGS	Load pointer using GS
LSS	Load pointer using SS
MOVSX	Move with sign extended
MOVZX	Move with zero extended
POPAD	Pop all double (32 bit) registers
POPD	Pop double register
POPFD	Pop double flag register
PUSHAD	Push all double registers
PUSHD	Push double register
PUSHFD	Push double flag register

Additional 80486 instruction

BSWAP	Byte swap

Additional Pentium instruction

MOV	Move to/from control register

Instruction types

Arithmetic instructions

8088 instruction set

AAA	ASCII adjust for addition
AAD	ASCII adjust for division
AAM	ASCII adjust for multiply
AAS	ASCII adjust for subtraction
ADC	Add byte or word plus carry
ADD	Add byte or word
CBW	Convert byte or word
CMP	Compare byte or word
CWD	Convert word to double-word
DAA	Decimal adjust for addition
DAS	Decimal adjust for subtraction
DEC	Decrement byte or word by one
DIV	Divide byte or word
IDIV	Integer divide byte or word
IMUL	Integer multiply byte or word
INC	Increment byte or word by one
MUL	Multiply byte or word (unsigned)
NEG	Negate byte or word
SBB	Subtract byte or word and carry (borrow)
SUB	Subtract byte or word

Additional 80386 instructions

CDQ	Convert double-word to quad-word
CWDE	Convert word to double-word

Additional 80486 instructions

CMPXCHG	Compare and exchange
XADD	Exchange and add

Additional Pentium instruction

CMPXCHG8B	Compare and exchange 8 bytes

Instruction types

Bit manipulation instructions

8088 instruction set

AND	Logical AND of byte or word
NOT	Logical NOT of byte or word
OR	Logical OR of byte or word
RCL	Rotate left trough carry byte or word
RCR	Rotate right trough carry byte or word
ROL	Rotate left byte or word
ROR	Rotate right byte or word
SAL	Arithmetic shift left byte or word
SAR	Arithmetic shift right byte or word
SHL	Logical shift left byte or word
SHR	Logical shift right byte or word
TEST	Test byte or word
XOR	Logical exclusive-OR of byte or word

Additional 80386 instructions

BSF	Bit scan forward
BSR	Bit scan reverse
BT	Bit test
BTC	Bit test and complement
BTR	Bit test and reset
BTS	Bit test and set
SETcc	Set byte on condition
SHLD	Shift left double precision
SHRD	Shift right double precision

Instruction types

String instructions

8088 instruction set

CMPS	**Compare byte or word string**
LODS	**Load byte or word string**
MOVS	**Move byte or word string**
MOVSB(MOVSW)	**Move byte string (word string)**
REP	**Repeat**
REPE (REPZ)	**Repeat while equal (zero)**
REPNE (REPNZ)	**Repeat while not equal (not zero)**
SCAS	**Scan byte or word string**
STOS	**Store byte or word string**