جَامعة المَنارة
MANARA UNIVERSITY

كلية الهندسة المعلوماتية
بنيان حواسيب 2
الفصل الصيفي 2024-2025
المحاضرة الثانية

د كنده أبو قاسم

8086 Architecture:

Features of 8086

- t is a 16-bit Microprocessor (µp).It's ALU, internal registers works with 16bit binary word.
- 8086 has a 20 bit address bus can access up to $2^{20} = 1\ \text{MB}$ memory locations.
- 8086 has a 16bit data bus. It can read or write data to a memory/port either 16bits or 8 bit at a time.
- It can support up to 64K I/O ports.
- It provides 14, 16 -bit registers.
- Frequency range of 8086 is 6-10 MHz
- It has multiplexed address and data bus AD0- AD15 and A16 – A19.

It can prefetch up to 6 instruction bytes from memory and queues them in order to speed up instruction execution

- It requires +5V power supply.

Architecture البنيان

المكونات الأساسية للمعالج 8086

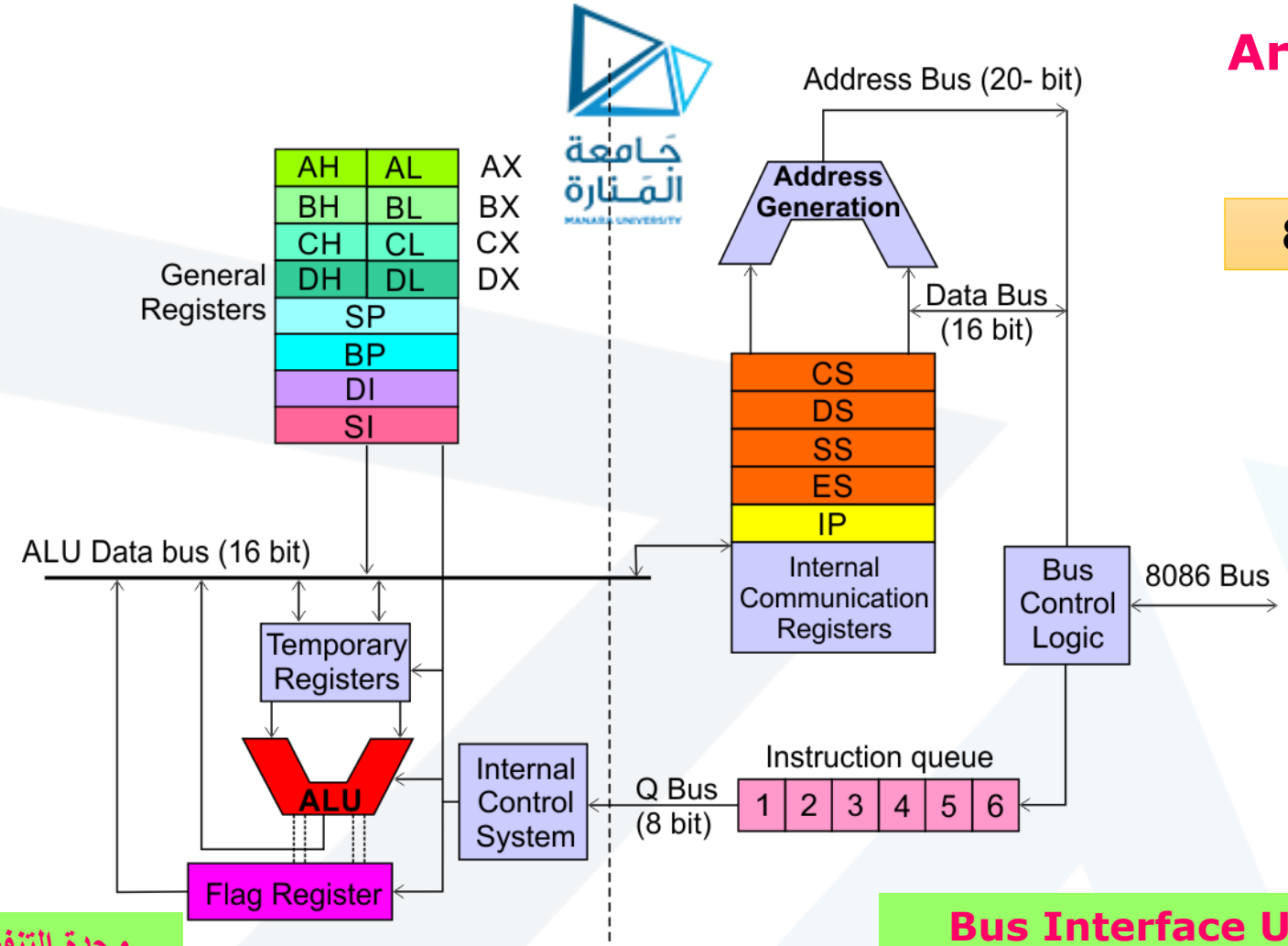
وحدة الباص البينية Bus Interface Unit (BIU)

وحدة التنفيذ Execution Unit (EU)

العنوان الفيزيائي physical addresses

Architecture البنيان

المكونات الأساسية للمعالج 8086



Execution Unit (EU) وحدة التنفيذ

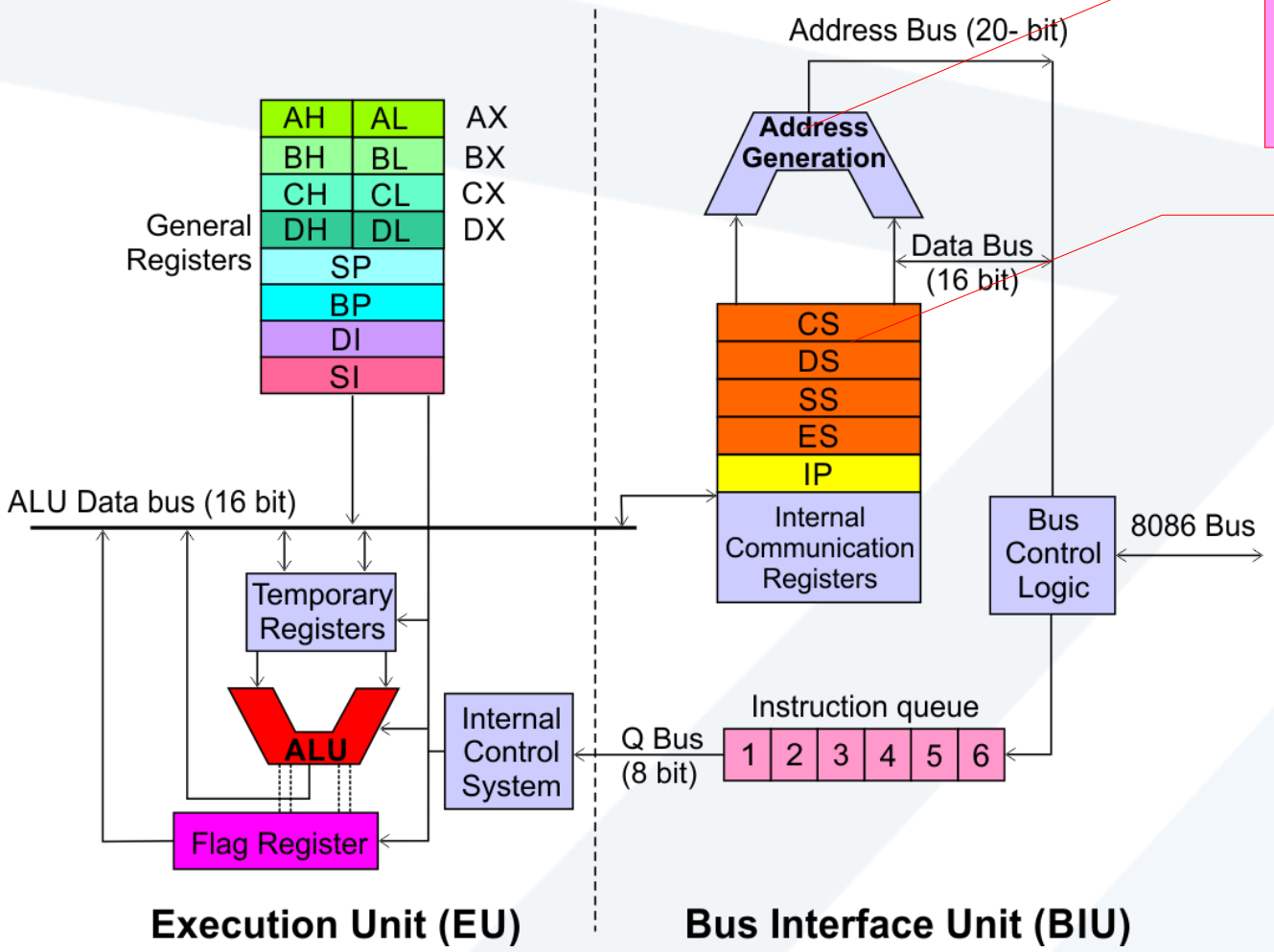
تنفذ التعليمات التي تم جلبها مسبقاًمن قبل الـBIU.

تعمل الوحدتان بشكل ستقل وظيفياً.

Bus Interface Unit (BIU) وحدة الباص البينية

تقوم وحدة الباص البينية بجلب اتعليمات وقراءة البيامات من ذاكرة البيانات ومن بوابات الإدخال/الإخراج، وتكتب البيانات إلى الذاكرة وبوابات الإدخال/ الإخراج.

Architecture



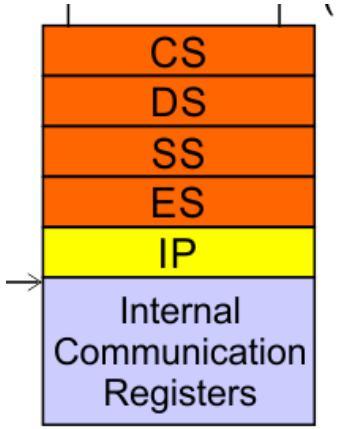
جامع مخصص لتوليد العناوين بعرض 20 bit

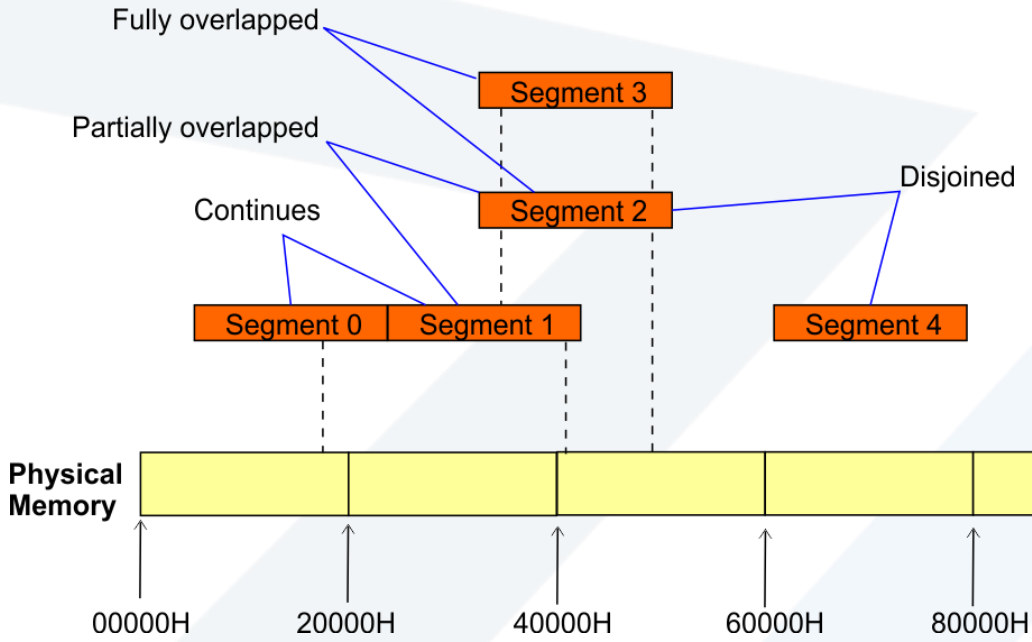
أربع مسجلات قطاعات كل من بعرض 16-bit

Code Segment (CS)
Data Segment (DS)
Stack Segment (SS)
Extra Segment (ES)

Bus Interface Unit (BIU)

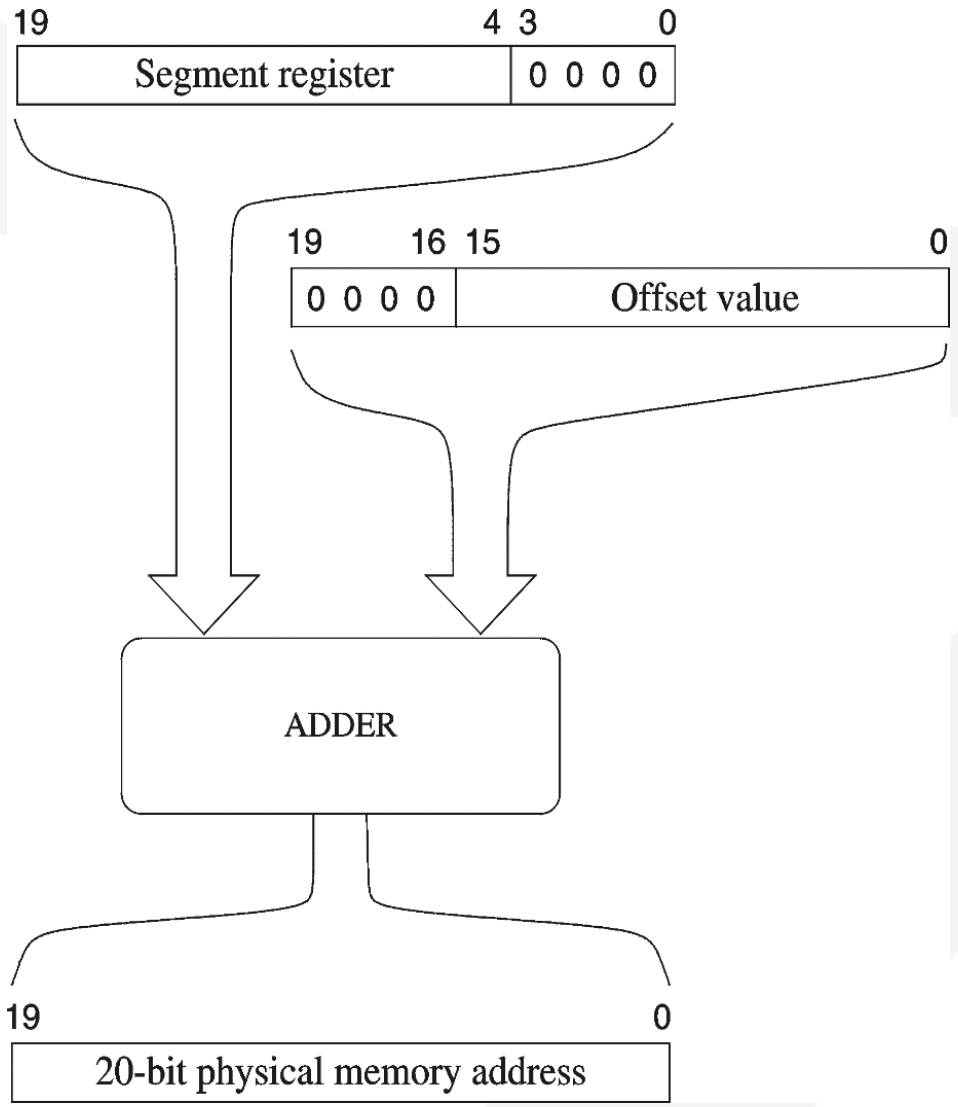
Segment Registers





- يمكن للبرنامج الوصول إلى البيانات والشيفرة في مختف القطاعات من خلال تغيير محتوى مسجل القطاع ليشير إلى القطاع المرغوب.
- يمكن للـ 8086 أن يعنون مباشرةً حتى أربع قطاعات (256 K bytes ضمن فضاء العناوين البالغ 1 M) في لحظة ما.
- تقسم الـ1 ميكا بايت الخاصة بالـ 8086's إلى قطاعات كل منها 64K bytes.

حساب العنوان الفيزيائي



- Conversion from logical to physical addresses

لحساب العنوان الفيزيائي نحتاج الى

CS:IP

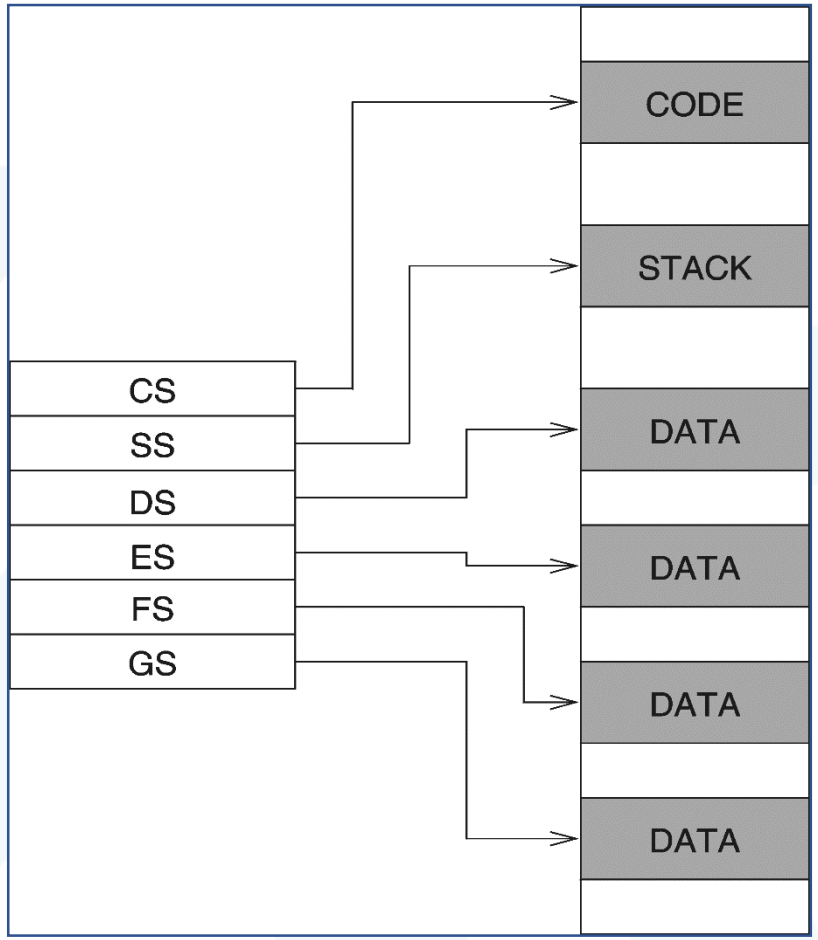
CS يشير على عنوان المقطع

IP يشير على مقدار الازاحة عن بداية المقطع Offset

```
11000  (add 0 to base)
+  450  (offset)
 11450  (physical address)
```

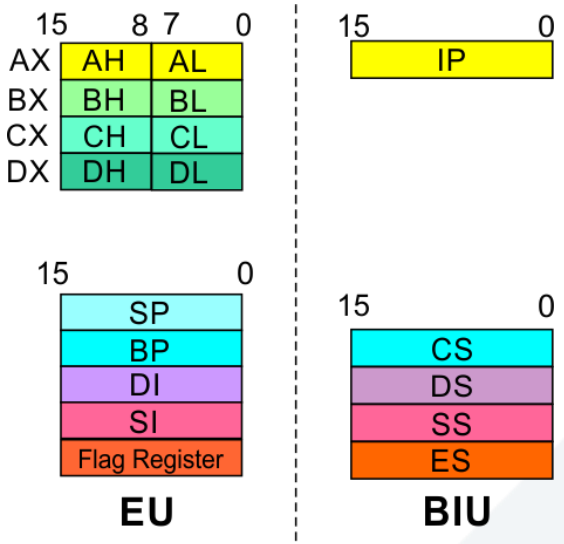
Real Mode Architecture (cont'd)

- Programs can access up to six segments at any time
- Two of these are for
  - Data
  - Code
- Another segment is typically used for
  - Stack
- Other segments can be used for
  - data, code,..



Bus Interface Unit (BIU)

Segment Registers

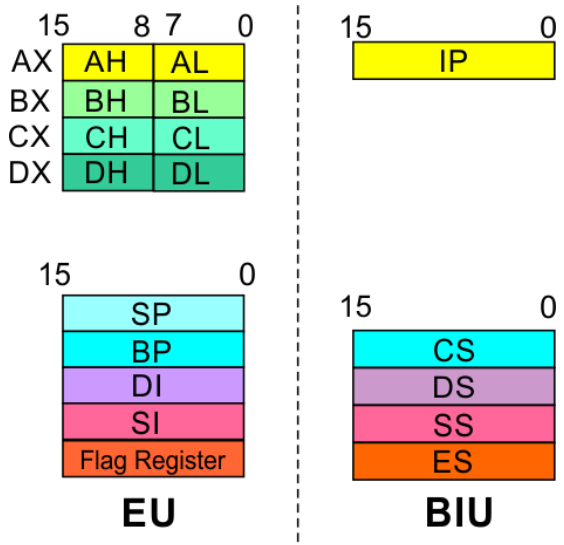


مسجل قطاع الشيفرة Code Segment Register

- 16-bit
- يحتوي CS على أساس أوبداية قطاع الشيفرة الحالي؛ بينما يحتوي IP على المسافة أو الإزاحة من هذا العنوان إلى البايت التي تبدأ بها التعليمة التالية التي سيتم جلبها.
- تحسب الـ BIU العنوان الفيزيائي المكون من 20-bit من العنوان المنطقي بإزاحة محتوى CS 4 خانات نحو اليسار ومن ثم جمع محتوى الـIP المكون من 16 خانة ثنائية إليه.
- وهذا يعني إزاحة جميع تعليمات البرنامج بقيمة تحسب من ضرب محتوى المسجل CS بـ 16 وإضافته إلى الإزاحة المقدمة من خلال القيمة المخزنة في IP.

Bus Interface Unit (BIU)

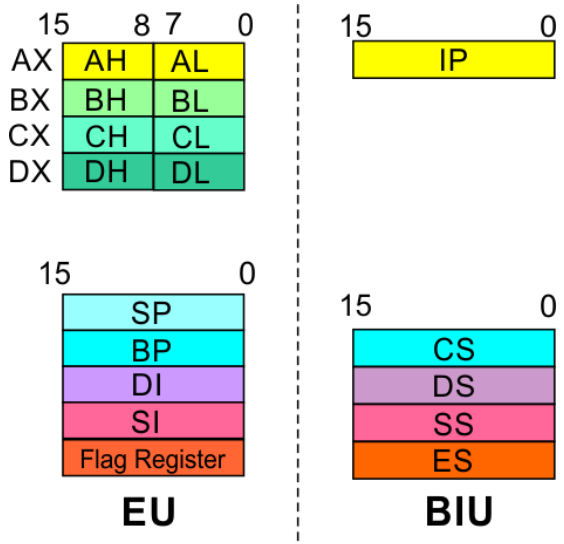
مسجل قطاع البيانات Data Segment Register



- 16-bit
- يشير إلى قطاع البيانات الحالي؛ ويتم جلب الحدود لمعظم التعليمات من خلال هذا القطاع.
- يستخدم كإزاحة المحتوى لمكون من 16-bit لدليل المصدر Source Index (SI) أو لدليل لهدف Destination Index (DI) أو إزاحة من 16-bit من أجل حساب العنوان الفيزيائي المكون من 20-bit.

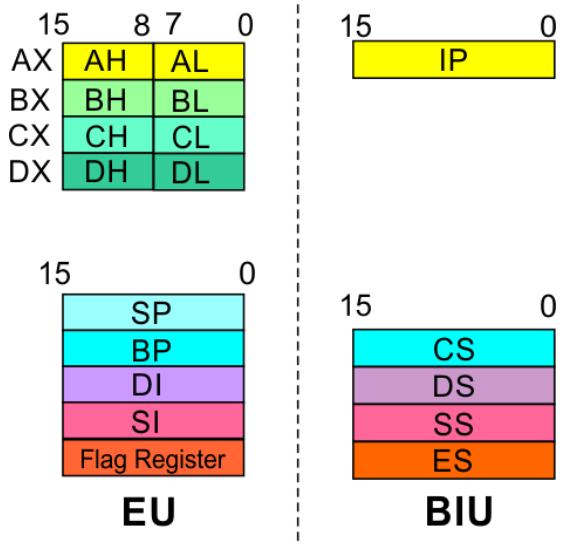
مسجال القطاع الإضافي Extra Segment Register

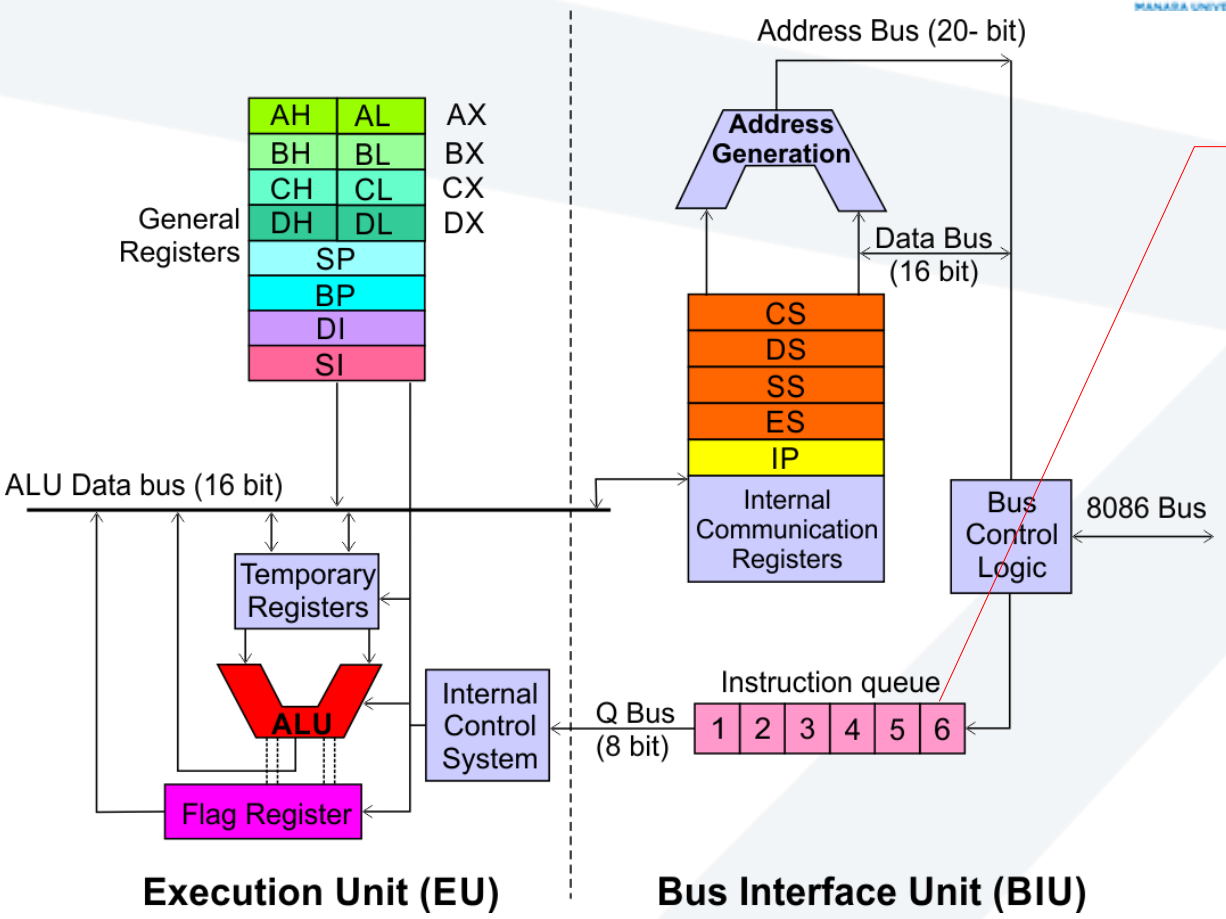
- 16-bit
- يشير إلى القطاع الإضافي الذي يشير إلى البيانات (إضافةً إلى 64K المشار إليها بالـDS).
- تعليمات السلاسل String instructions التي تستخدم ES و DI لتحديد العنوان الفيزيائي المكون من 20-bit للسلسلة الهدف.



مؤشر التعليمة Instruction Pointer

- 16-bit
- يشير دوماً إلى التعليمة التالية الواجب تنفيذها ضمن قطاع الشيفرة التنفيذي الحالي. **currently executing code segment.**
- وبالتالي فإن هذا امسجل يحتوي على إزاحة العنوان المكون من **16-bit** **offset address** والذي يشير إلى شيفرة التعليمة التالية ضمن قطاع الشيفرة والبالغ حجمه **64Kb.**
- تجري مزايدة محتوى هذا المسجل عند حصول تنفيذ التعليمة التالية.



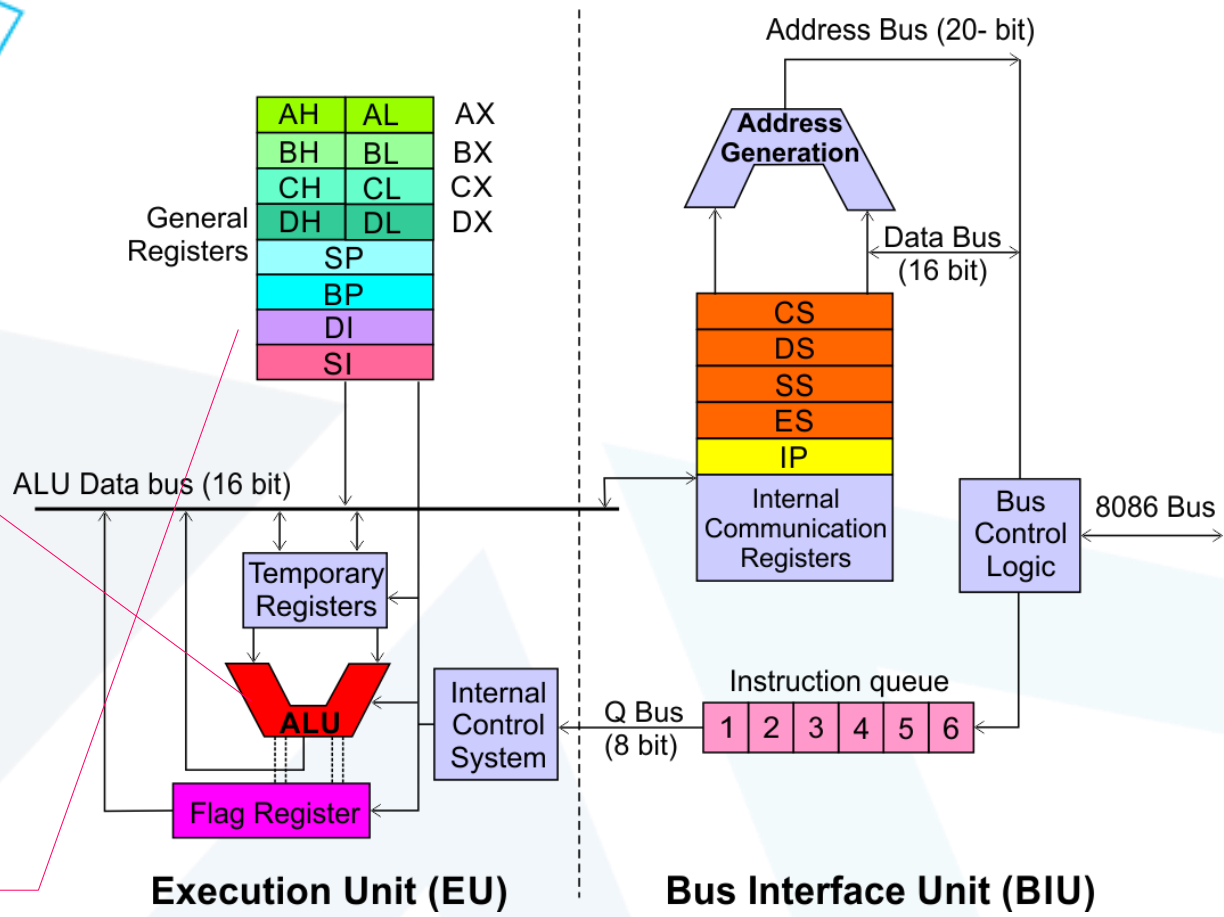


Instruction queue رتل التعليمة

- مجموعة من البايتات تعمل بمبدأ الداخل الأول هو الخارج الأول First-In-First-Out (FIFO) يمكن أن نستخدمها للجلب المسبق لـ 6-bytes من شيفرة التعليمة من الذاكرة
- نقوم بذلك لنسرّع التنفيذ من خلال التراكب الزمني لعمليتي جلب التعليمة وتنفيذها.
- الآلية المعروفة بتوازي السير المتحرك pipelining.

تقوم **EU** بفك ترميز وتنفيذ التعليمات

يقوم مفككك ترميز موجود في وحدة التحكم
بالنظام **EU control system**
بترجمة التعليمات



وحدة **ALU** بعرض **16-bit** لإنجاز العمليات الحسابية والمنطقية

أربع مسجلات أغراض عامة **(AX, BX, CX, DX);**

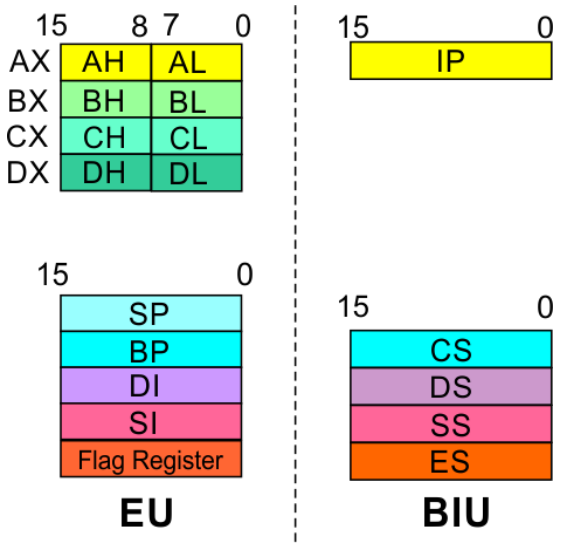
مسجلات المؤشرات **(Stack Pointer, Base Pointer);**
و
Index registers (Source Index, Destination Index)
كل منها بعرض **16-bits**

يمكننا استخدام بعض المسجلات بعرض **16 bit** كمسجلين بعرض **8 bit** مثل:

AX can be used as AH and AL
BX can be used as BH and BL
CX can be used as CH and CL
DX can be used as DH and DL

البنيانArchitecture

EU Registers



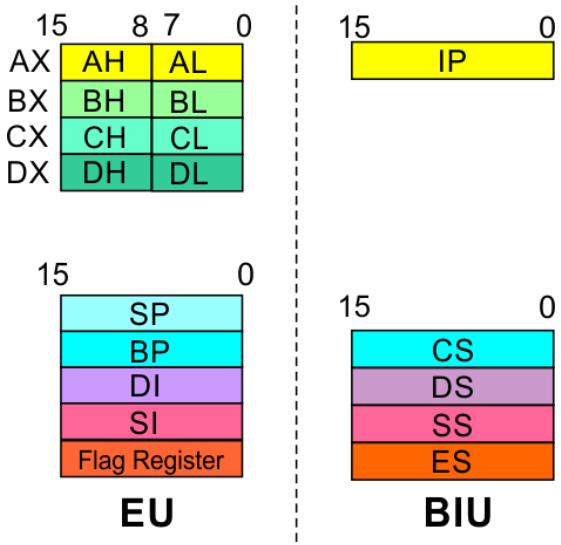
Execution Unit (EU)

المسجل المراكم أو المجمع Accumulator Register (AX)

- يتألف من مسجلين كل منهما سعة **8-bit** هما **AL** و **AH**، ويمكننا أن نضمهما إلى بعضهما البعض ليشكلا مسجلاً بسعة **16-bit** هو المسجل **AX.**
- يحوي **AL** في هذه الحالة البايت الأقل أهمية من الكلمة **WORD** بينما يحتوي **AH** على البايت الأكثر أهمية من الكلمة.
- تستخدم تعليمات **I/O** المسجلات **AX** أو **AL** من أجل إدخال/إخراج **16** أو **8 bit** من البيانات من أو إلى **I/O port.**
- كذلك الأمر فإن الضرب والقسمة يستخدمان **AX** أو **AL.**

Execution Unit (EU)

EU Registers

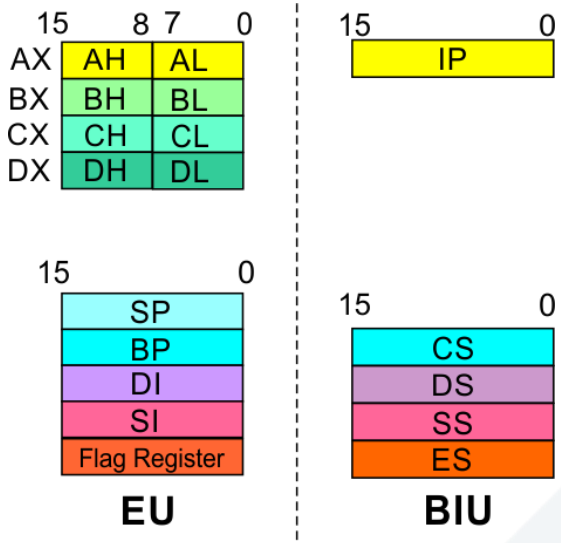


Base Register (BX) المسجل الأساس

- يتألف من مسجلين كل منهما سعة **8-bit** هما **BL** و **BH**، ويمكننا أن نضمهما إلى بعضهما البعض ليشكلا مسجلاً بسعة **16-bit** هو المسجل **BX**.
- يحوي **BL** في هذه الحالة البايت الأقل أهمية من الكلمة **WORD** بينما يحتوي **BH** على البايت الأكثر أهمية من الكلمة.
- وهو مسجل أغراض عامة يمكن استخدام محتواه من أجل عنونة الذاكرة **8086 memory**.
- جميع مرجعيات الذاكرة التي تستخدم محتوى هذا المسجل للعنونة (الإزاحة) تستخدم القطاع **DS** كعنوان أساس.

Execution Unit (EU)

EU Registers



Counter Register (CX) مسجل العد

- يتألف من مسجلين كل منهما سعة **8-bit** هما **CL** و **CH**، ويمكننا أن نضمهما إلى بعضهما البعض ليشكلا مسجلاً بسعة **16-bit** هو المسجل **CX**.

- يحوي **CL** في هذه الحالة البايت الأقل أهمية من الكلمة **WORD** بينما يحتوي **CH** على البايت الأكثر أهمية من الكلمة.

- تعليمات مثل **SHIFT** و **ROTATE** و **LOOP** تستخدم محتوى المسجل **CX** كعداد **counter**.

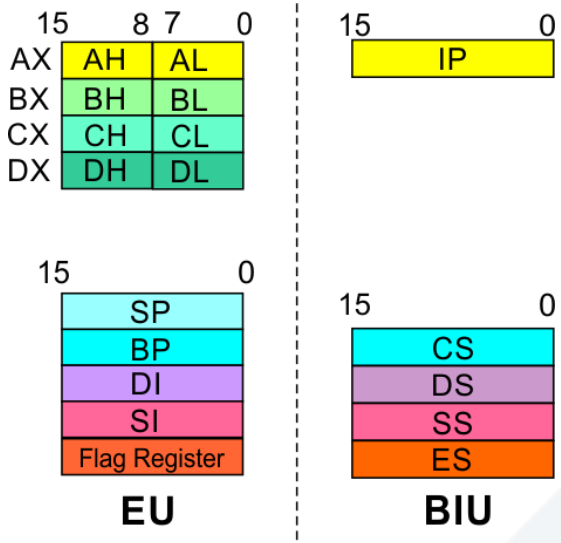
مثال:

التعليمة **LOOP START** تناقص آلياً المسجل **CX** بمقدار **1** بدون التأثير على الأعلام وتختبر فيما إذا أصبح **[CX] = 0**.

فإذا كان مساوياً للصفر فإن الـ **8086** ينفذ التعليمة التالية؛ وإلاّ فإنه يتفرع إلى العلاّم **label START**.

Execution Unit (EU)

EU Registers



مسجل البيانات Data Register (DX)

- يتألف من مسجلين كل منهما سعة **8-bit** هما **DL** و **DH**، ويمكننا أن نضمهما إلى بعضهما البعض ليشكلا مسجلاً بسعة **16-bit** هو المسجل **DX**.

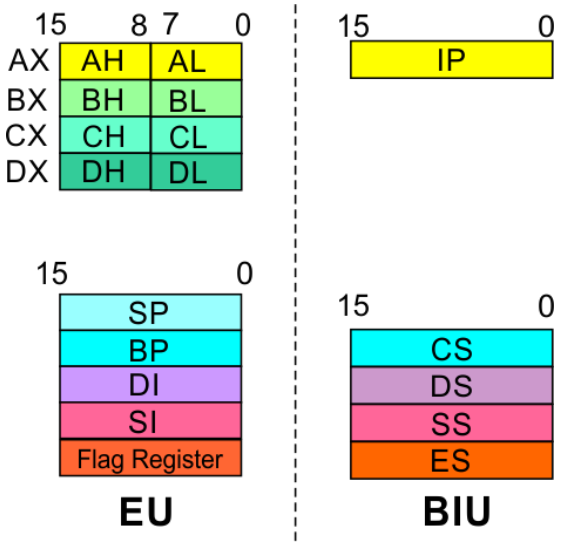
- يحوي **DL** في هذه الحالة البايت الأقل أهمية من الكلمة **WORD** بينما يحتوي **DH** على البايت الأكثر أهمية من الكلمة.

مؤشر

- يستخدم للاحتفاظ بالـ**16-bit** العليا من النتيجة في حال عمليات ضرب **16×16** أو الـ**16-bit** العليا من المقسوم قبل إجراء قسمة **32÷16** كما يحتفظ بالباقي **16-reminder** بعد عملية القسمة.

Execution Unit (EU)

EU Registers

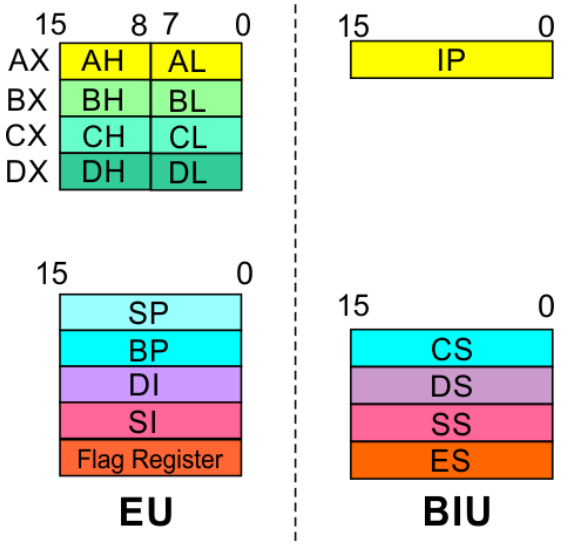


Stack Pointer (SP) and Base Pointer (BP) مؤشر الكدسة ومؤشر الأساس

- يستخدم **SP** و **BP** للوصول إلى البيانات في قطاع الكدسة.**stack segment** .
- يستخدم **SP** كإزاحة عن قطاع الكدسة الحالي أثناء تنفيذ التعليمة التي تستخدم قطاع الكدسة في الذاكرة الخارجية
- يجري تحديث محتوى الـ **SP** آلياً (مزايدة أو مناقصة) نتيجة تنفيذ تعليمتي **POP** أو **PUSH**.
- يحتوي **BP** على إزاحة العنوان لـ**SS** الحالي والذي يستخدم من خلال التعليمات التي تستخدم نمط عنونة الأساس **based addressing mode**.

Execution Unit (EU)

EU Registers



دليل المصدر و دليل الهدف Source Index (SI) and Destination Index (DI)

- يستخدم في العنونة الدليلية indexed addressing.
- تستخدم التعليمات التي تعالج سلاسل البيانات المسجلين SI و DI بالإسهام مع القطاعين DS و ES على التوالي وذلك للتمييز بين عناوين المصدر والهدف.

FLAG REGISTERV مسجل الأعلام

علم الحمل Carry Flag

يأخذ قيمة الواحد عندما يكون هنالك حمل من الخانة الأكثر أهمية في عمليات الجمع أو استعارة في عمليات الطرح

علم الحمل المساعد Auxiliary Carry Flag

يأخذ قيمة الواحد إذا كان هنالك حمل أو منقول من الرباعية الأولىlowest nibble، الخانة الثالثة إلى الرابعة أثناء الجمع أواستعارة إلى الخانة الثالثة أثناء الطرح.

علم الإشارة Sign Flag

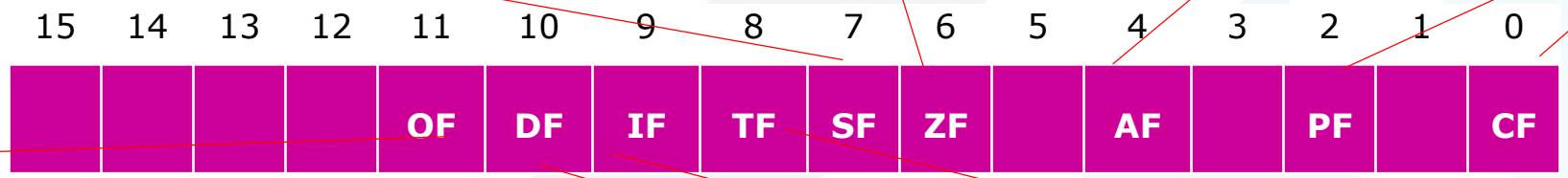
يأخذ هذا العلم قيمة الواحد عندما تكون نتيجة أية عملية حسابية سالبة.

علم الصفر Zero Flag

ياخذ قيمة الواحد إذا كانت نتيجة الحساب أو المقارنة المنجزة بالتعليمة صفرية.

علم الإنجابية Parity Flag

يأخذ قيمة الواحد إذا احتوت البايت الدنيا من النتيجة على عدد زوجي من الواحدات. ويأخذ قييمة الصفر في حال الاحتواء على عدد فردي.



علم الطفح Over flow Flag

يأخذ قيمة الواحد إذا حصل طفح أي إذا كانت نتيجة العملية ذات الإشارة بحجم يكفي لاستيعابه في المسجل الهدف. إذا كانت النتيجة أكبر من 7 خانات في حال استخدام مسجل 8 خانات للنتيجة أو 15 خانة في حال استخدمنا مسجل بسع 16 خانة في العمليات ذات الإشارة،

علم المصيدة Tarp Flag

إذا كانت قيمة هذا العلم واحد فإن المعالج يدخل في نمط تنفيذ الخطوة الواحدية من خلال توليد مقاطعة داخلية بعد تنفيذ كل تعليمة.

علم الاتجاه Direction Flag

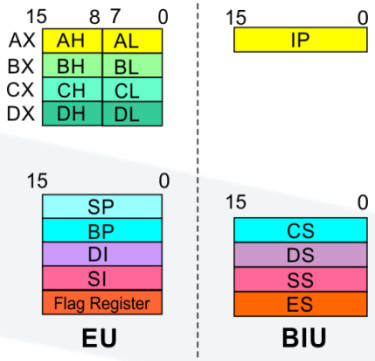
يستخدم في عمليات معالجة السلاسل. فإذا كان DF=0 تعالج السلسلة ابتداءً من العنوان الأصغر وحتى العنوان الأكبر ضمن نمط مزايدة آلي. أما إذا كانت DF=1 فسيجري معالجة السلسلة من من العنوان الأعلى إلى الأصغر ضمن نمط مناقصة آلي.

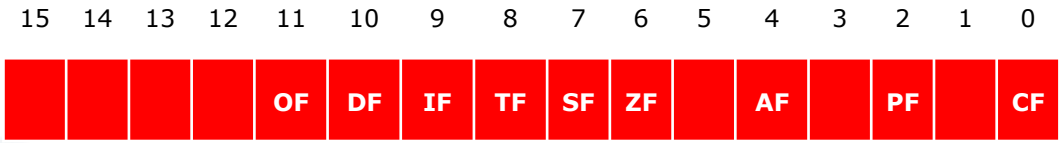
علم المقاطعة Interrupt Flag

في حال أخذ هذا العلم قيمة الواحد يسبب تمكين المعالج من تمييز المقاطعات الخارجية القابلة للحجب وفي حال جرى تصفيره فإنه يلغي هذا التمكين.

Architecture البنيان

يمكن تصنيف مسجلات المعالج 8086 في أربع مجموعات





Sl.No.	Type	Register width	Name of register
1	مسجلات أغراض عامة General purpose register	16 bit	AX, BX, CX, DX
		8 bit	AL, AH, BL, BH, CL, CH, DL, DH
2	مسجلات المؤشرات Pointer register	16 bit	SP, BP
3	المسجلات الدليلية Index register	16 bit	SI, DI
4	مؤشر التعليمة Instruction Pointer	16 bit	IP
5	مسجلات القطاعات Segment register	16 bit	CS, DS, SS, ES
6	الأعلامFlag (PSW)	16 bit	Flag register

Status Flags

Carry Flag (CF): After performing any operation on an 8086 microprocessor, the carry flag will only be set if a carry is created from the MSB of the result.

Parity Flag (PF): the number of 1s in the binary data determines parity. Two different kinds of parity exist:

Even Parity: The state in which all of the binary data's 1s are even.

When the binary data has an odd number of 1s, it is said to have odd parity.

If there is even parity in the data following the execution of the instruction, the PF is set for the flag. The flag is reset if not.

Auxiliary Carry Flag (AF): In an arithmetic operation, when the carry is generated from bit D3 to D4, the auxiliary carry flag is set to 1. (Starting from bit D0)

Zero Flag (ZF): The zero flags are set to 1 if an appropriate operation (either logical or arithmetic) on the instructions yields a zero result. If not, it stays reset.

Sign Flag (SF): The sign flag is set to 1 if the result of any arithmetic or logic operation carried out in the provided instruction is negative. If not, the sign flag is left reset in the event of a favorable outcome.

Overflow Flag (OF): After every arithmetic or logic operation, this flag will be set if the register overflows with data. This occurs when the carry is received in MSB but the carried out bit cannot be stored in the register.

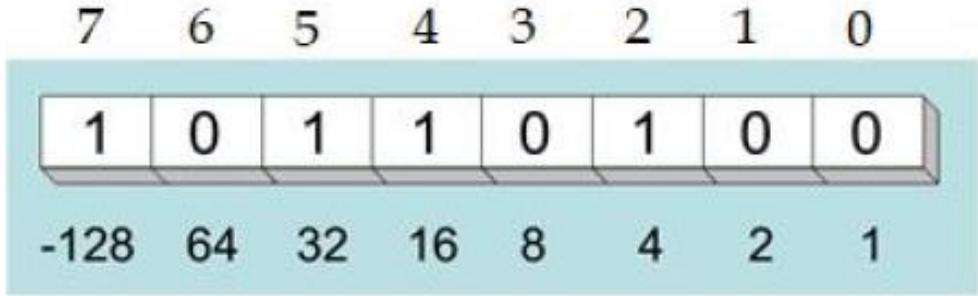
Example Addition 0Fh + 08h

AL=0Fh	0	0	0	0	1	1	1	1
BL=08h	0	0	0	0	1	0	0	0
AL+BL=17h CF = 0	0	0	0	1	0	1	1	1

CF=0, OF=0

Both operands and the result are positive. PF=1, ZF=0, SF=0, AF=1

Unsigned numb (decimal)	Its binary form	Its hex form	Signed numb (decimal)
0	00000000	00	+0
1	00000001	01	+1
2	00000010	02	+2
...	...	...	
127	01111111	7F	+127
128	10000000	80	-128
129	10000001	81	-127
...	...	...	...
254	11111110	FE	-2
255	11111111	FF	-1



Reminder of the second complement

Range of represented values: -2^{n-1} *to* $2^{n-1}-1$

If n=8, then the range is -127 to +127

Example : Addition 0Fh + F8h

AL=0Fh	0	0	0	0	1	1	1	1
BL=F8h	1	1	1	1	1	0	0	0
AL+BL=07h CF = 1	0	0	0	0	0	1	1	1

CF=1, OF=0, Operands have different sign bits.

PF=0, ZF=0, SF=0, AF=1

Control Flags

To guide the microprocessor for certain tasks, the control flags are employed. A control flag can be of three types:

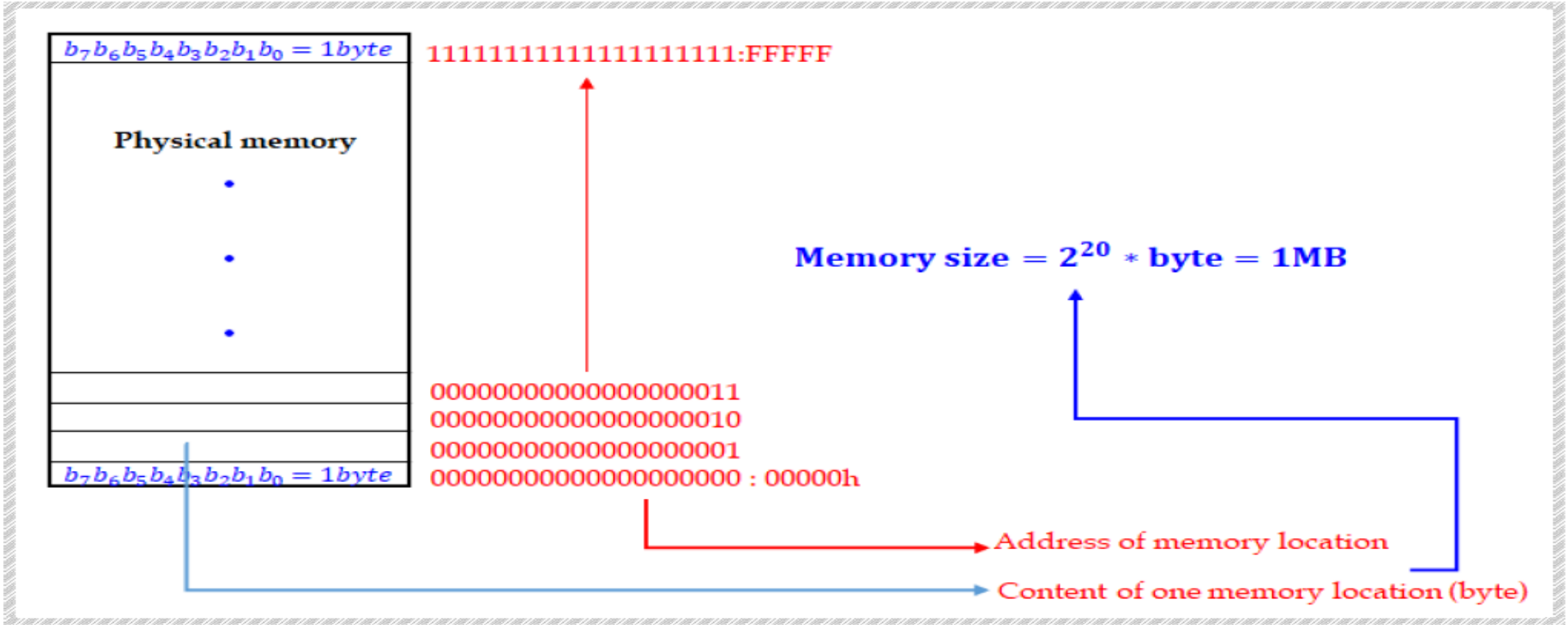
Trap Flag (TF): This flag is used of we need single-step debugging in our code. If the TF is set, then the execution will be done step by step. Otherwise, the free-running operation will be done.

Interrupt Flag (IF): This flag is used to enable the Interrupt. The microprocessor is capable of handling interrupts only if this flag is in the set mode. Otherwise, any interrupt raised while the execution of the instructions will not be handled by the microprocessor.

Direction Flag (DF): This flag is used for string operations. If this flag is set, the string will be read from higher-order bits to lower order bits and vice versa

Memory segmentation and physical address

The 8086 has 20 address bits, so it can address 2 bytes or 1 MB. The address of the first memory location is 0000 0000 0000 0000 0000 (00000h), the last case is 1111 1111 1111 1111 1111 1111 (FFFFFh).



Architecture البنيان

Segment	Starting Address	Last Address	Segment Pointer
Segment 0 (like page 0)	00000	0FFFF	00000
Segment 1 (like page 1)	10000	1FFFF	10000
Segment 2 (like page 2)	20000	2FFFF	20000
.....	.....	.....	.....
Segment 14 (like page 14)	E0000	EFFFF	E0000
Segment 15 (like page 15)	F0000	FFFFF	F0000

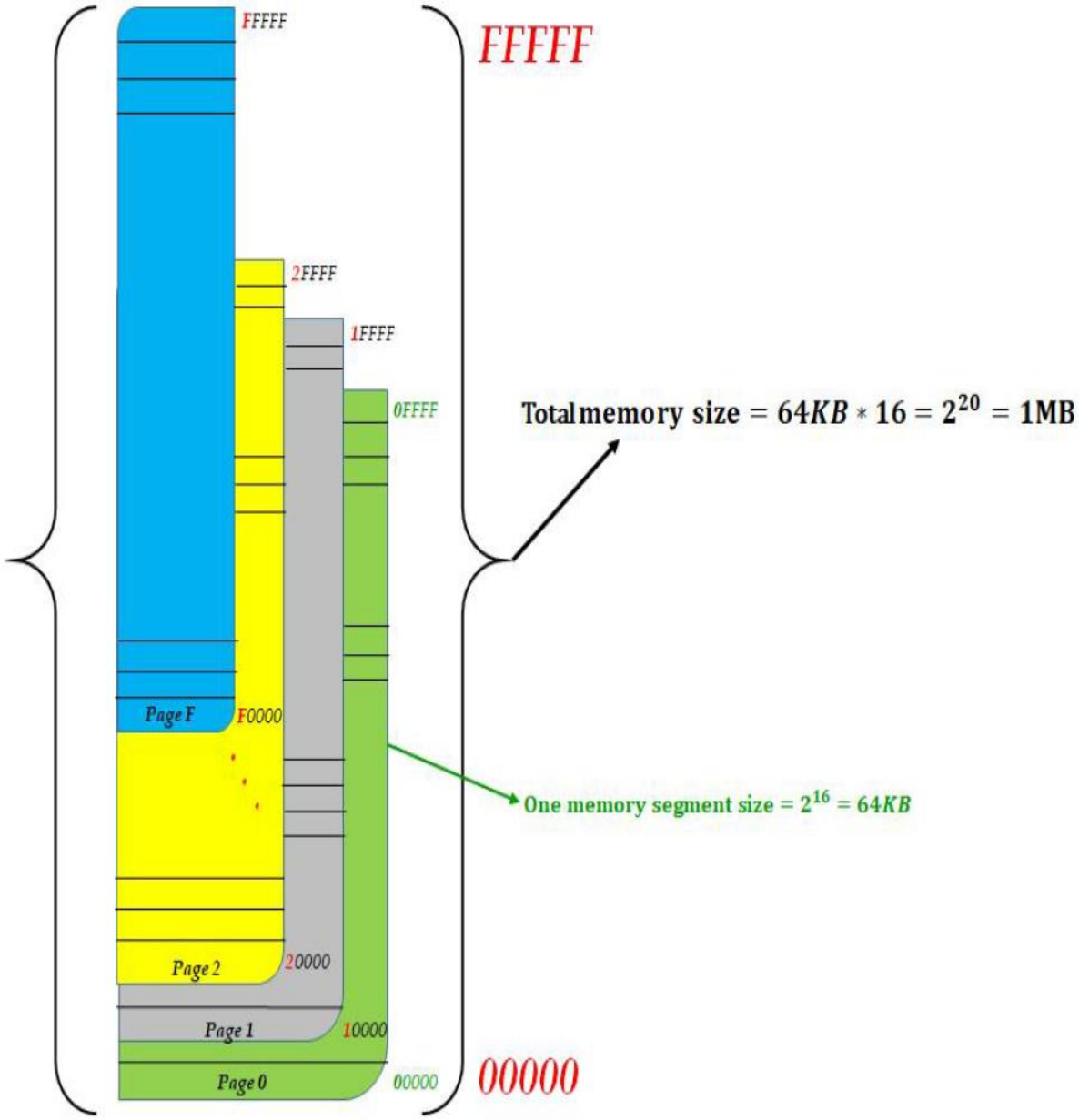
4 bit	4 bit	4 bit	4 bit	0000	Segment Value

+

4 bit	4 bit	4 bit	4 bit	Offset Value

=

4 bit	4 bit	4 bit	4 bit	4 bit	Physical Address = 20 bit



Answer

a. Logical address is CS:IP = 2500:95F3

b. The physical address that will be sent to the address bus by the CPU of 8086 microprocessor is:

Physical Address (20bits) = CS * 10 + IP = 25000 +95F3 = ⋯h

If CS = 24F6h and IP = 634Ah, determine.

a. The logical address in the code segment
b. The offset address in the code segment
c. The physical address in the 1 MB memory
d.The lower range in the code segment
e. The upper range in the code segment

Addressing in a specific Segment

To execute a program, the 8086 fetches the instructions from the code segment.

The logical address of an instruction consists in CS (code segment) and IP (the instruction pointer)

The logical address is then

$$\text{Logical Address} = \text{CS: IP}$$

If CS=2500h and IP=95F3h

a. What is the logical address in the 64 KB segment?

b. What will be the physical address in the 1MO memory?

Example

If CS = 24F6h and IP = 634Ah, determine.

a. The logical address in the code segment
b. The offset address in the code segment
c. The physical address in the 1 MB memory
d.The lower range in the code segment
e. The upper range in the code segment

Answer

a. The logical address is : 24F6:634A

b. The offset address is 634A

c. The physical address is 24F6*10+634A = 2B2AA (20 bits)

d.The lower range in the code segment is (24F6:0000) = 24F6+0000=24F6

- The lower range in the physical memory is 24F60+0000=24F60 (20 bits)

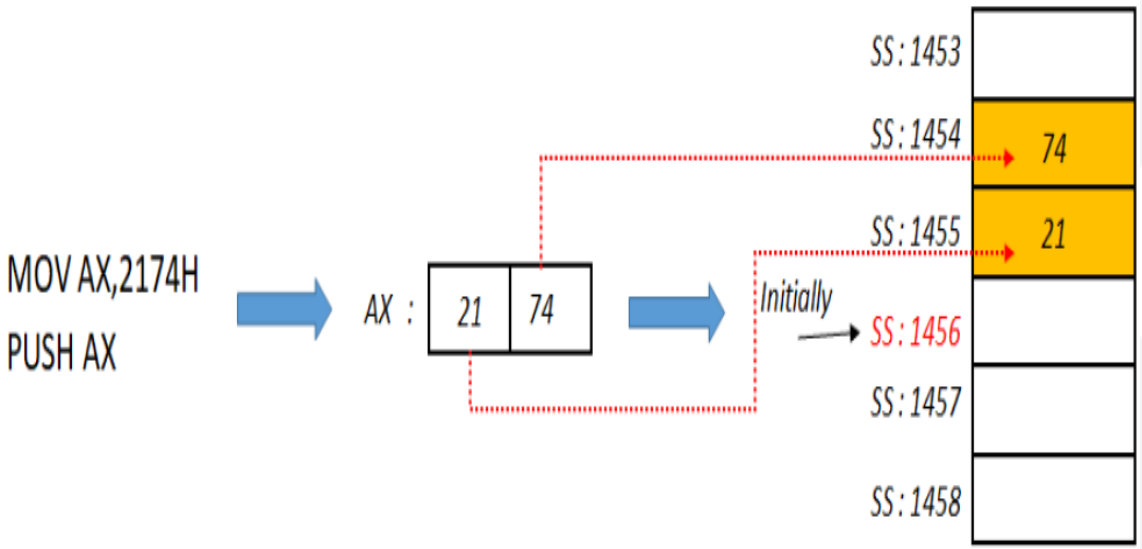
e. The upper range in the code segment is (24F6: FFFF) = 24F6+FFFF=24F6

- The upper range in the physical memory is 24F60+FFFF=34F5F (20 bits)

Stack Segment and push-pop concept

Example 1

After the following instruction is executed, what are the contents of AX, the top of the stack, and SP given that SP=1456H?



AX=2174H (stays the same), SP=1454H (decremented by 2)

To access the stack in the memory, SP (stack pointer) and SS (stack segment) must be loaded. All CPU registers, with the exception of segment registers and SP, can be loaded from and transferred to the stack

A push is the process of storing the CPU register in the stack.

PUSH source ;

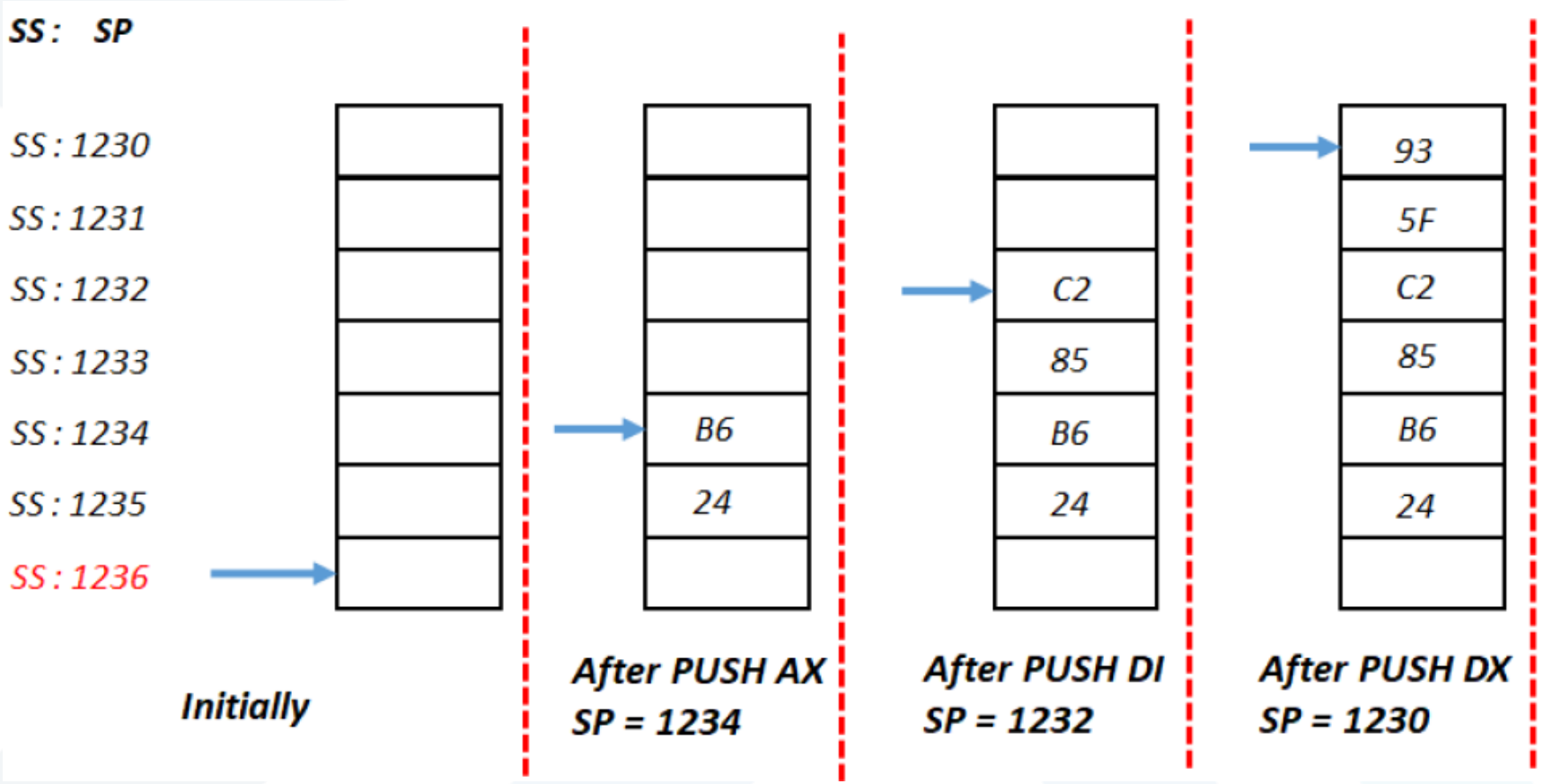
Mnemonic *Operand*

- *Copy the content of the source (16 bits) into the stack*
- *SP register is decremented by 2*

Example

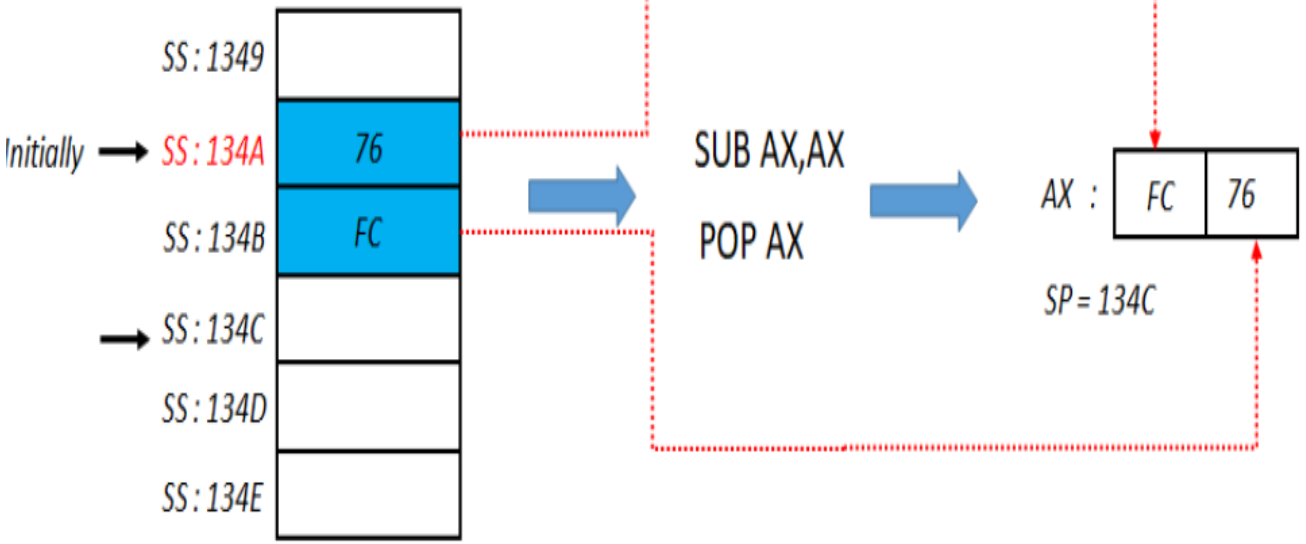
You are given SP=1236H, AX=24B6H, DI=85C2H, and DX=5F93H, show the contents of the stack and SP as each of the following instructions is executed.

POP AX
POP DI
POP DX



Assume that SP=134AH and the illustration on the left shows the content of the top of the stack.
What will be the content of AX and SP after the execution of the following instructions?

SUB AX, AX
POP AX



Note:

- SUB AX, AX makes AX empty before transferring data
- After popping data from SS to the register AX. SS will be empty

Popping operation in Stack

It is the opposite procedure of Push instruction
Loading the contents of the stack into the CPU register is called a pop.

POP destination ;

Mnemonic *Operand*

Copy the top of the stack into destination (16-bit register)

SP register is incremented by 2