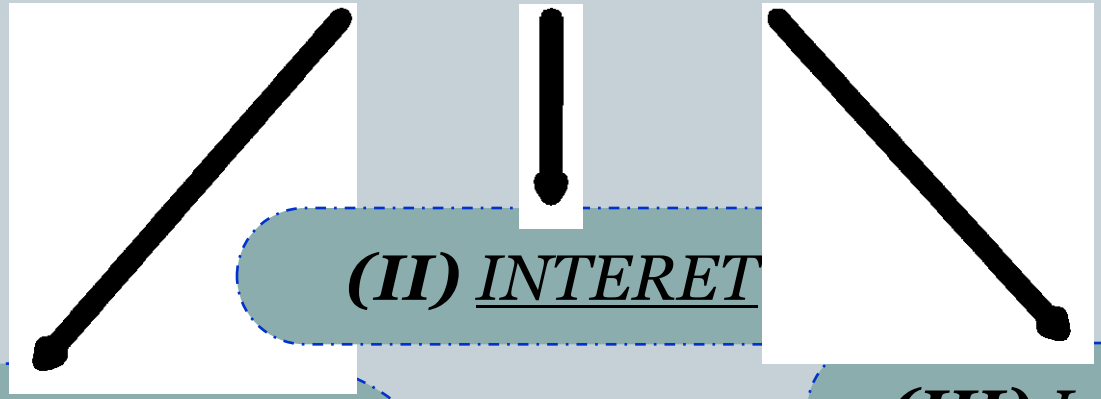
ARCHITECTURE DES ORDINATEURS

6)- Introduction à l' ASSEMBLEUR 'x86'

Biblio :: **1/« *Assembly Langage for INTEL-based computers*»**
[Kip R. IRVINE] – Ed. Prentice Hall, 1999 – ISBN: 0-13-660390-4.
2/« *An assembly langage introduction to computer architecture* (*using the intel pentium)*»
[K. MILLER] – Ed. Oxford University Press, 1999 – ISBN: 0-19-512376-X
3/«*The Intel microprocessors : Architecture, Programming & Interfacing*»
[Barry B. BREY] – Ed. Prentice Hall 2006 – ISBN: 0-13-119506-9.
4/ «*The Hardware Bible*»
[Winn L. Rosch] – Ed. QUE /McMillan computer Publishing – ISBN: 0-7897-1743-3.
@ web::
1-http://css.csail.mit.edu/6.858/2013/readings/
2-http://www.ustudy.in/node/

6)- Introduction à l'ASM 'x86'

Objectif :: *A S S E M B L E U R ('ASM')*



(II) *INTERET*

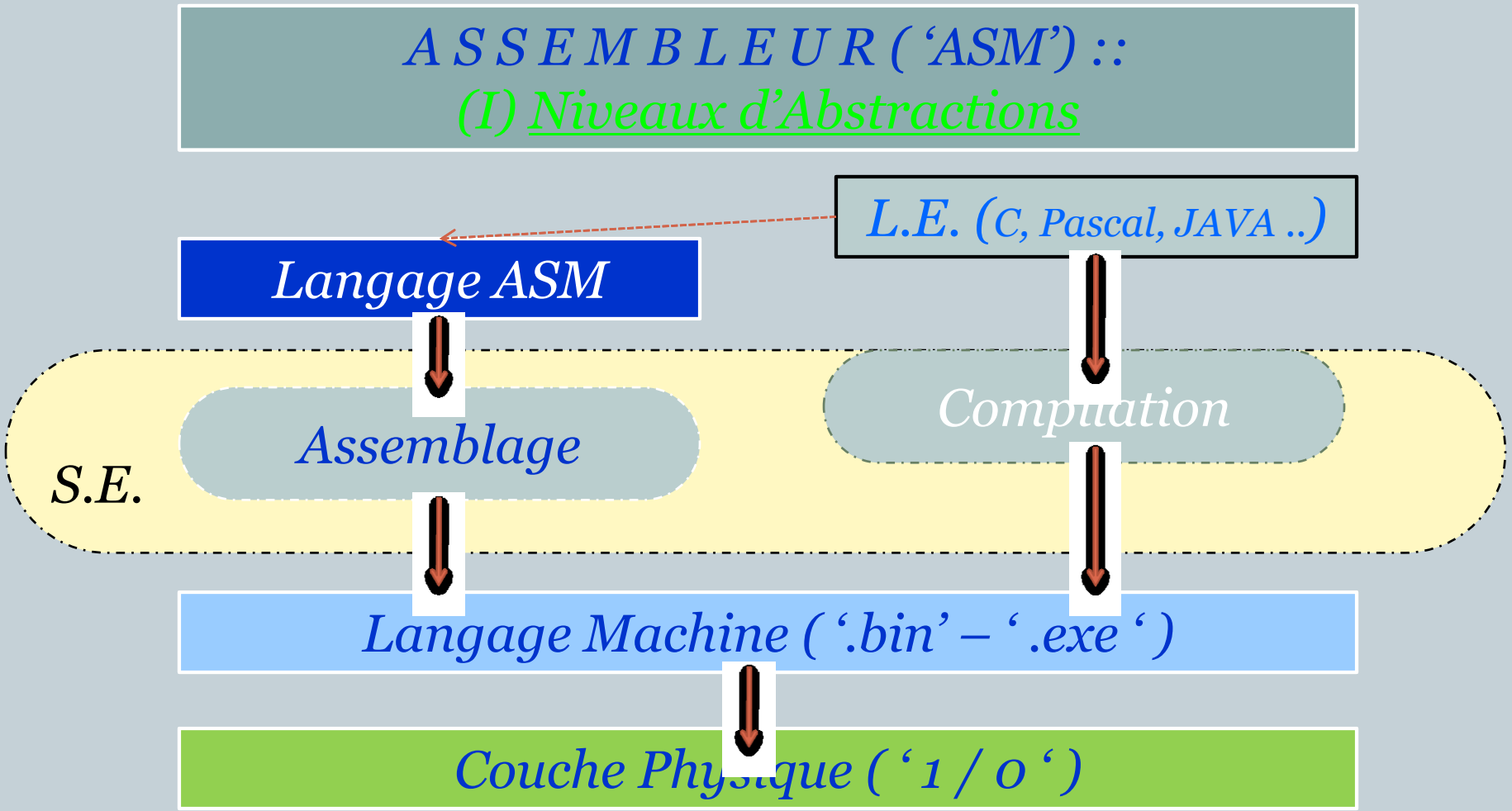
(I) *Points d'accès ou* ***NIVEAUX D'ABSTRACTION***

(III) *LANGAGE*

* *LEXIQUE*

* *SYNTAXE*

6)- Introduction à l'ASM 'x86'



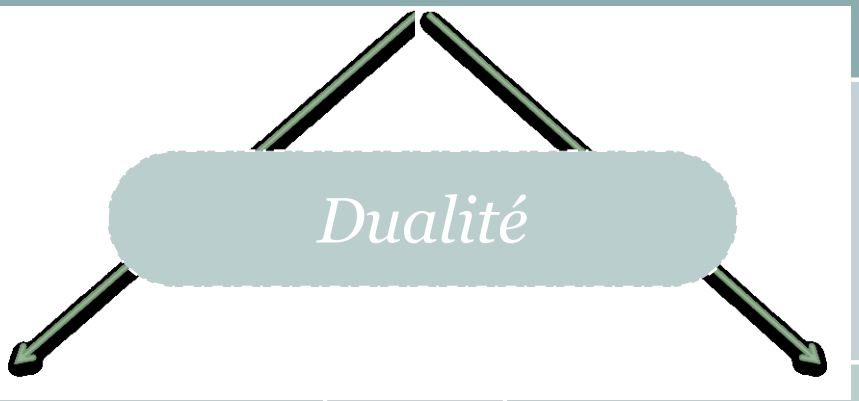
6)- Introduction à l'ASM 'x86'

A S S E M B L E U R ('ASM') ::
(II) I n t é r ê t

1)-	*Accès aux ressources matérielles 'Low Level'*
2)-	*Programmation optimale orientée 'hardware'*
3)-	*Exploitation optimale des ressources matérielles*
4)-	*Applications diverses : 'systèmes embarqués', 'pilotage process non-standard', etc ..*
5)-	*Accès à la bibliothèque 'système' : Fonctions DOS & BIOS .. (récupère toute la puissance du SE)*

6)- Introduction à l'ASM 'x86'

A S S E M B L E U R ('ASM') ::
(III) L e L a n g a g e

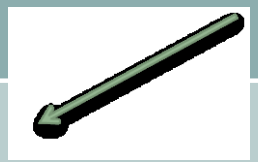


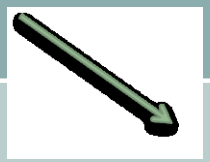
LANGAGE de programmation

ENVIRONNEMENT (programmation)

6)- Introduction à l'ASM 'x86'

A S S E M B L E U R ('ASM') ::
(III) L e L a n g a g e





LANGAGE

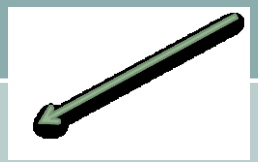
Lexique ::
J.I. orienté 'CPU'

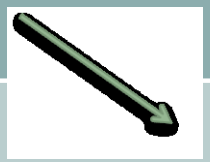
ENVIRONNEMENT

EDI. // 'SE'

6)- Introduction à l'ASM 'x86'

A S S E M B L E U R ('ASM') ::
(III) L e L a n g a g e

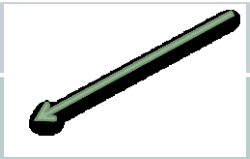


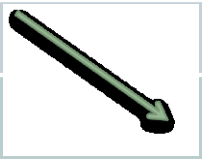


LANGAGE	*ENVIRONNEMENT*
Lexique :: *J.I. orienté 'CPU'*	*EDI. // 'SE'*
'Chaque CPU a son propre JI'	*'Exécution orientée S.E.'*

6)- Introduction à l'ASM 'x86'

A S S E M B L E U R ('ASM') ::
(III) L e L a n g a g e

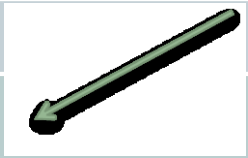




LANGAGE	*ENVIRONNEMENT*
J.I. orienté 'CPU'	*EDI. // 'SE'*
'80286' :: Jeu de 120 instructions :: Transfert, Arithmétique, Logique, Décalage, Condition, ...	*DOS / WINDOWS/UNIX :: DEBUG, TASM, MASM, **NASM**, Chrome, Easy code, ...*

6)- Introduction à l'ASM 'x86'

A S S E M B L E U R ('ASM') ::
(III) L e L a n g a g e



LANGAGE

SYNTAXE orientée 'Ressources_CPU'

6)- Introduction à l'ASM 'x86'

L A N G A G E // SYNTAXE ::

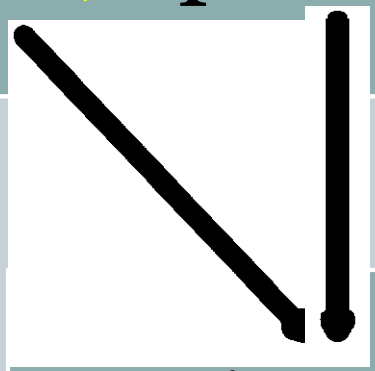
'Modèle générique d'1 instruction'

[**CODOP** oper_1, oper_2;]

6)- Introduction à l'ASM 'x86'

[**CODOP** oper_1, oper_2;]

*Mnémonique // **CODe** de l'**OP**ération*



Opérandes *:: Registre, Immédiat, Variable, @_RAM, @_ E/S*

6)- Introduction à l'ASM 'x86'

[**CODOP** oper_1, oper_2;]

(Destination & 'Source') (Source)

*Mnémonique // **CODe** de l'**OP**ération*

Operandes *:: Registre, Immédiat, Variable, @_RAM, @_ E/S*

Règle PERMANENTE

Oper_1 : 'Destination' & 'Source'
Oper_2 : 'Source'

6)- Introduction à l'ASM 'x86'

M O D E L E ...

[**CODOP** oper_1, oper_2;]

```
; illustration: " Type d'opérandes "

CLC ;          "Mnémonique" / 'sans op'
NOP ;          "Mnémonique" / 'sans op'
INC AX ;       "Mnémonique" + '1 opérande'
ADD AX, 2 ;    "Mnémonique" + '2 ops (Reg, Im)'
ADD AX, BX ;   "Mnémonique" + '2 ops (Reg, Reg)'
```

6)- Introduction à l'ASM 'x86'

EXEMPLE/ILLUSTRATION…

Instructions élémentaires::
(1) Transfert
(2) Arithmétique
(3) Logique

6)- Introduction à l'ASM 'x86'

```
; exemple 1= instructions & typologie

MOV AX, 10 ;          (1)
SHL AX, 2 ;           (2)
MOV BX, AX ;          (3)
ADD AX, 2 ;           (4)
SUB AX, 0011b ;       (5)
MUL AX, 0Ah ;         (6)
AND AX, 00FFh ;       (7)
OR AX, 0081h ;        (8)
```

6)- Introduction à l'ASM 'x86'

```
; exemple 1= instructions & typologie

MOV AX, 10 ;          (1) TRANSFERT
SHL AX, 2 ;           (2) DECALAGE / ARITHM.
MOV BX, AX ;          (3) TRANSFERT
ADD AX, 2 ;           (4) ARITHMETIQUE
SUB AX, 0011b ;       (5) ARITHMETIQUE
MUL AX, 0Ah ;         (6) ARITHMETIQUE
AND AX, 00FFh ;       (7) LOGIQUE
OR AX, 0081h ;        (8) LOGIQUE
```

6)- Introduction à l'ASM 'x86'

```
; exemple 2= manipulation Reg d'@dressage

      MOV AX, 10 ;          (1)
      MOV SI, AX ;          (2)
      MOV BX, AX ;          (3)
      MOV DI, SI+100 ;      (4)
      ADD AX, [BX] ;        (5)
      MOV [DI], [SI+100] ; (6)
```

6)- Introduction à l'ASM 'x86'

	ACTION	RESULTS
; exemple 2=		
;manipulation Reg		
;d'@dressage		
MOV AX, 10 ; (1)	AX ← 10	AX = 10
MOV SI, AX ; (2)	SI ← AX	SI = 10
MOV BX, AX ; (3)	BX ←AX	BX = 10
MOV DI, SI+100 ; (4)	DI ← SI+100	DI = 110
ADD AX, [BX] ; (5)	AX ← [BX]	AX = [@=10]
MOV [DI], [SI]+100; (6)	@(DI) ←[SI]+100	[@=110] =[@=10]+100