

# Microprocesseurs (MIC)

## Chap. 7 : Le co-processeur Intel 8087

# Sommaire

- 1 Introduction
- 2 Le calcul par pile
- 3 Intel 8087
  - Pile et registres
  - Instructions

# Les co-processeurs

- Co-processeur : circuit destiné à ajouter une fonction à un processeur classique.
  - Co-processeurs arithmétiques (calcul en virgule flottante)
  - Co-processeurs graphiques (2D, 3D)
  - Co-processeurs spécialisés dans le chiffrement
- But : augmenter les performances de la machine pour un type de calcul précis.

# Le co-processeur Intel 8087

- Co-processeur arithmétique pour la gamme x86
- x86 : uniquement calculs de nombres entiers
  - Calculs de réels simulés avec des entiers
  - Ex :  $2.5 + 3.4 = (2 + 3).(5 + 4)$
- Le 8087 introduit des instructions de calcul de nombres réels

# Le co-processeur Intel 8087

- Disponible dès le 8086, comme circuit intégré séparé
- Versions ultérieures : 80187, 80287, 80387
- À partir du 80486DX (32 bits), le co-processeur est intégré dans la même puce que le processeur «classique»



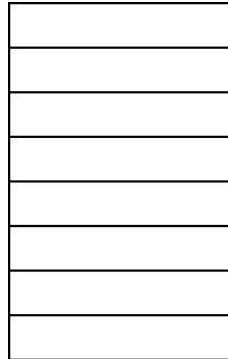
Figure: Processeur 80386 avec co-processeur 80387 séparé

# Sommaire

- 1 Introduction
- 2 Le calcul par pile
- 3 Intel 8087
  - Pile et registres
  - Instructions

# Le calcul par pile

- Calcul par pile



# Le calcul par pile

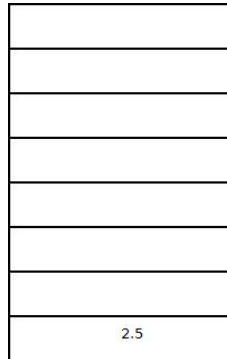
- Calcul par pile
- Exemple 1 :  $2.5 + 3.3$





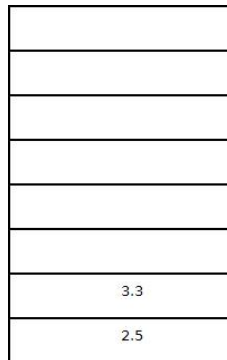
# Le calcul par pile

- Calcul par pile
- Exemple 1 :  $2.5 + 3.3$ 
  - PUSH 2.5



# Le calcul par pile

- Calcul par pile
- Exemple 1 :  $2.5 + 3.3$ 
  - PUSH 2.5
  - PUSH 3.3



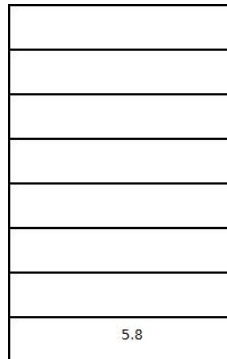
# Le calcul par pile

- Calcul par pile
- Exemple 1 :  $2.5 + 3.3$ 
  - PUSH 2.5
  - PUSH 3.3
  - SOMME

5.8

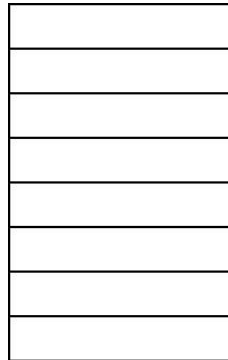
# Le calcul par pile

- Calcul par pile
- Exemple 1 :  $2.5 + 3.3$ 
  - PUSH 2.5
  - PUSH 3.3
  - SOMME
- Le résultat final est au fond de la pile



# Le calcul par pile

- Exemple II :  
 $(2.5 + 3.3) \times (1.2 + 0.8)$



# Le calcul par pile

- Exemple II :  
 $(2.5 + 3.3) \times (1.2 + 0.8)$ 
  - PUSH 2.5



# Le calcul par pile

- Exemple II :  
 $(2.5 + 3.3) \times (1.2 + 0.8)$ 
  - PUSH 2.5
  - PUSH 3.3

3.3
2.5

# Le calcul par pile

- Exemple II :  
 $(2.5 + 3.3) \times (1.2 + 0.8)$ 
  - PUSH 2.5
  - PUSH 3.3
  - SOMME

5.8



# Le calcul par pile

- Exemple II :  
 $(2.5 + 3.3) \times (1.2 + 0.8)$ 
  - PUSH 2.5
  - PUSH 3.3
  - SOMME
  - PUSH 1.2

1.2
5.8

# Le calcul par pile

- Exemple II :  
 $(2.5 + 3.3) \times (1.2 + 0.8)$ 
  - PUSH 2.5
  - PUSH 3.3
  - SOMME
  - PUSH 1.2
  - PUSH 0.8

0.8
1.2
5.8

# Le calcul par pile

- Exemple II :  
 $(2.5 + 3.3) \times (1.2 + 0.8)$ 
  - PUSH 2.5
  - PUSH 3.3
  - SOMME
  - PUSH 1.2
  - PUSH 0.8
  - SOMME

2.0
5.8

# Le calcul par pile

- Exemple II :  
 $(2.5 + 3.3) \times (1.2 + 0.8)$ 
  - PUSH 2.5
  - PUSH 3.3
  - SOMME
  - PUSH 1.2
  - PUSH 0.8
  - SOMME
  - PRODUIT

11.6

# Le calcul par pile

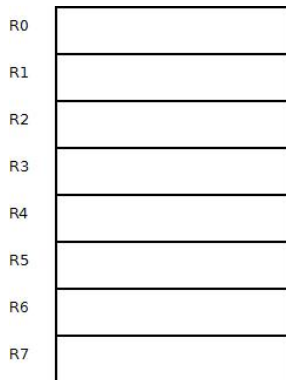
- Principe
  - Les opérandes sont pushées sur la pile
  - Les calculs poppent les opérandes et pushent le résultat
  - Le résultat final est disponible au fond de la pile
- Technique très employée
  - Compilateurs
  - Certains processeurs et co-processeurs
  - **Co-processeur Intel 8087**
  - ...

# Sommaire

- 1 Introduction
- 2 Le calcul par pile
- 3 Intel 8087**
  - Pile et registres
  - Instructions

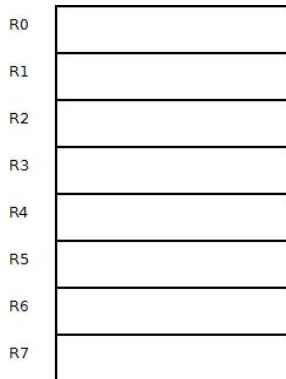
# La pile du 8087 : registres

- Pile hardware dans le co-processeur



# La pile du 8087 : registres

- Pile hardware dans le co-processeur
- 8 registres de 80 bits : *R0* à *R7*





# La pile du 8087 : registres

- Pile hardware dans le co-processeur
- 8 registres de 80 bits : *R0* à *R7*
- Valeurs possibles dans les registres :

R0	
R1	
R2	
R3	
R4	
R5	
R6	
R7	

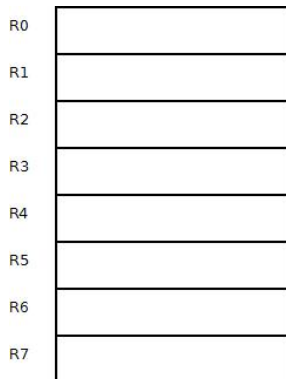
# La pile du 8087 : registres

- Pile hardware dans le co-processeur
- 8 registres de 80 bits : *R0* à *R7*
- Valeurs possibles dans les registres :
  - Nombre réel (80 bits)

R0	
R1	
R2	
R3	
R4	
R5	
R6	
R7	

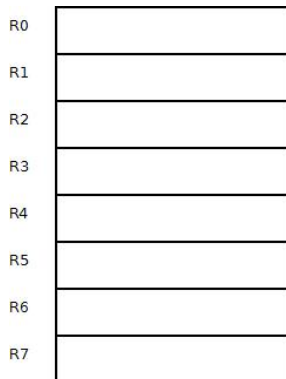
# La pile du 8087 : registres

- Pile hardware dans le co-processeur
- 8 registres de 80 bits : *R0* à *R7*
- Valeurs possibles dans les registres :
  - Nombre réel (80 bits)
  - « vide »



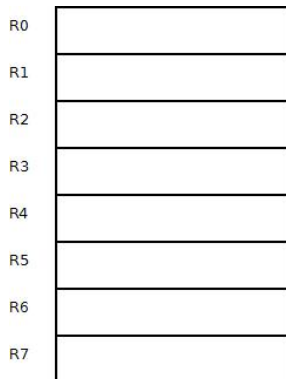
# La pile du 8087 : registres

- Pile hardware dans le co-processeur
- 8 registres de 80 bits : *R0* à *R7*
- Valeurs possibles dans les registres :
  - Nombre réel (80 bits)
  - « vide »
  - « infini »



# La pile du 8087 : registres

- Pile hardware dans le co-processeur
- 8 registres de 80 bits : *R0* à *R7*
- Valeurs possibles dans les registres :
  - Nombre réel (80 bits)
  - « vide »
  - « infini »
  - « NAN »



# La pile du 8087 : utilisation

- Les registres  $R0$  à  $R7$  ne sont pas employés tel quels

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	2.0	← st0
R6	3.3	← st1
R7	2.5	← st2

# La pile du 8087 : utilisation

- Les registres *R0* à *R7* ne sont pas employés tel quels
- Raccourcis de notation : *st0* à *st7*

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	2.0	← st0
R6	3.3	← st1
R7	2.5	← st2

# La pile du 8087 : utilisation

- Les registres *R0* à *R7* ne sont pas employés tel quels
- Raccourcis de notation : *st0* à *st7*
  - *st0* = sommet de pile

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	2.0	← st0
R6	3.3	← st1
R7	2.5	← st2



# La pile du 8087 : utilisation

- Les registres *R0* à *R7* ne sont pas employés tel quels
- Raccourcis de notation : *st0* à *st7*
  - *st0* = sommet de pile
  - *st1* = élément suivant

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	2.0	← st0
R6	3.3	← st1
R7	2.5	← st2

# La pile du 8087 : utilisation

- Les registres *R0* à *R7* ne sont pas employés tel quels
- Raccourcis de notation : *st0* à *st7*
  - *st0* = sommet de pile
  - *st1* = élément suivant
  - *st2* = élément suivant

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	2.0	← st0
R6	3.3	← st1
R7	2.5	← st2

# La pile du 8087 : utilisation

- Les registres *R0* à *R7* ne sont pas employés tel quels
- Raccourcis de notation : *st0* à *st7*
  - *st0* = sommet de pile
  - *st1* = élément suivant
  - *st2* = élément suivant
  - ...

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	2.0	← st0
R6	3.3	← st1
R7	2.5	← st2

# La pile du 8087 : exemple 1

- Pile vide au départ

R0	vide
R1	vide
R2	vide
R3	vide
R4	vide
R5	vide
R6	vide
R7	vide

# La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5

R0	vide
R1	vide
R2	vide
R3	vide
R4	vide
R5	vide
R6	vide
R7	2.5

← st0

# La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5
- PUSH 3.3

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	vide	
R6	3.3	← st0
R7	2.5	← st1

# La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5
- PUSH 3.3
- PUSH 2.0

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	2.0	← st0
R6	3.3	← st1
R7	2.5	← st2

# La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5
- PUSH 3.3
- PUSH 2.0
- PUSH 1.4

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	1.4	← st0
R5	2.0	← st1
R6	3.3	← st2
R7	2.5	← st3



# La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5
- PUSH 3.3
- PUSH 2.0
- PUSH 1.4
- PUSH 9.5

R0	vide	
R1	vide	
R2	vide	
R3	9.5	← st0
R4	1.4	← st1
R5	2.0	← st2
R6	3.3	← st3
R7	2.5	← st4

# La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5
- PUSH 3.3
- PUSH 2.0
- PUSH 1.4
- PUSH 9.5
- PUSH 2.7

R0	vide	
R1	vide	
R2	2.7	← st0
R3	9.5	← st1
R4	1.4	← st2
R5	2.0	← st3
R6	3.3	← st4
R7	2.5	← st5

# La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5
- PUSH 3.3
- PUSH 2.0
- PUSH 1.4
- PUSH 9.5
- PUSH 2.7
- PUSH 6.6

R0	vide	
R1	6.6	← st0
R2	2.7	← st1
R3	9.5	← st2
R4	1.4	← st3
R5	2.0	← st4
R6	3.3	← st5
R7	2.5	← st6

# La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5
- PUSH 3.3
- PUSH 2.0
- PUSH 1.4
- PUSH 9.5
- PUSH 2.7
- PUSH 6.6
- PUSH 3.9

R0	3.9	← st0
R1	6.6	← st1
R2	2.7	← st2
R3	9.5	← st3
R4	1.4	← st4
R5	2.0	← st5
R6	3.3	← st6
R7	2.5	← st7

## La pile du 8087 : exemple 2

- Pile remplie au départ

R0	3.9	← st0
R1	6.6	← st1
R2	2.7	← st2
R3	9.5	← st3
R4	1.4	← st4
R5	2.0	← st5
R6	3.3	← st6
R7	2.5	← st7

## La pile du 8087 : exemple 2

- Pile remplie au départ
- POP

R0	vide	
R1	6.6	← st0
R2	2.7	← st1
R3	9.5	← st2
R4	1.4	← st3
R5	2.0	← st4
R6	3.3	← st5
R7	2.5	← st6

## La pile du 8087 : exemple 2

- Pile remplie au départ
- POP
- POP

R0	vide	
R1	vide	
R2	2.7	← st0
R3	9.5	← st1
R4	1.4	← st2
R5	2.0	← st3
R6	3.3	← st4
R7	2.5	← st5

## La pile du 8087 : exemple 2

- Pile remplie au départ
- POP
- POP
- POP

R0	vide	
R1	vide	
R2	vide	
R3	9.5	← st0
R4	1.4	← st1
R5	2.0	← st2
R6	3.3	← st3
R7	2.5	← st4



## La pile du 8087 : exemple 2

- Pile remplie au départ
- POP
- POP
- POP
- POP

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	1.4	← st0
R5	2.0	← st1
R6	3.3	← st2
R7	2.5	← st3

## La pile du 8087 : exemple 2

- Pile remplie au départ
- POP
- POP
- POP
- POP
- POP

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	2.0	← st0
R6	3.3	← st1
R7	2.5	← st2

## La pile du 8087 : exemple 2

- Pile remplie au départ
- POP
- POP
- POP
- POP
- POP
- POP

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	vide	
R6	3.3	← st0
R7	2.5	← st1

## La pile du 8087 : exemple 2

- Pile remplie au départ
- POP
- POP
- POP
- POP
- POP
- POP
- POP

R0	vide
R1	vide
R2	vide
R3	vide
R4	vide
R5	vide
R6	vide
R7	2.5

← st0

## La pile du 8087 : exemple 2

- Pile remplie au départ
- POP
- POP
- POP
- POP
- POP
- POP
- POP
- POP

R0	vide
R1	vide
R2	vide
R3	vide
R4	vide
R5	vide
R6	vide
R7	vide

## La pile du 8087 : exemple 3

- $(2.5 + 3.3) \times (1.2 + 0.8)$

R0	vide
R1	vide
R2	vide
R3	vide
R4	vide
R5	vide
R6	vide
R7	vide

## La pile du 8087 : exemple 3

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- PUSH 2.5

R0	vide
R1	vide
R2	vide
R3	vide
R4	vide
R5	vide
R6	vide
R7	2.5

← st0

## La pile du 8087 : exemple 3

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- PUSH 2.5
- PUSH 3.3

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	vide	
R6	3.3	← st0
R7	2.5	← st1



## La pile du 8087 : exemple 3

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- PUSH 2.5
- PUSH 3.3
- SOMME

R0	vide
R1	vide
R2	vide
R3	vide
R4	vide
R5	vide
R6	vide
R7	5.8

← st0

## La pile du 8087 : exemple 3

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- PUSH 2.5
- PUSH 3.3
- SOMME
- PUSH 1.2

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	vide	
R6	1.2	← st0
R7	5.8	← st1

## La pile du 8087 : exemple 3

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- PUSH 2.5
- PUSH 3.3
- SOMME
- PUSH 1.2
- PUSH 0.8

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	0.8	← st0
R6	1.2	← st1
R7	5.8	← st2

## La pile du 8087 : exemple 3

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- PUSH 2.5
- PUSH 3.3
- SOMME
- PUSH 1.2
- PUSH 0.8
- SOMME

R0	vide	
R1	vide	
R2	vide	
R3	vide	
R4	vide	
R5	vide	
R6	2.0	← st0
R7	5.8	← st1

## La pile du 8087 : exemple 3

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- PUSH 2.5
- PUSH 3.3
- SOMME
- PUSH 1.2
- PUSH 0.8
- SOMME
- PRODUIT

R0	vide
R1	vide
R2	vide
R3	vide
R4	vide
R5	vide
R6	vide
R7	11.6

← st0

## La pile du 8087 : exemple 3

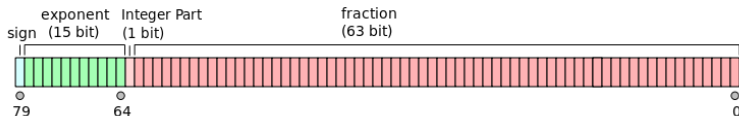
- $(2.5 + 3.3) \times (1.2 + 0.8)$
- PUSH 2.5
- PUSH 3.3
- SOMME
- PUSH 1.2
- PUSH 0.8
- SOMME
- PRODUIT
- Le résultat est dans *st0*

R0	vide
R1	vide
R2	vide
R3	vide
R4	vide
R5	vide
R6	vide
R7	11.6

← *st0*

# L'encodage des réels sous Intel 8087

- Les registres *st0* à *st7* contiennent des réels de 80 bits
- Encodage des réels en format « *x86 Extended Precision* »
  - Bit de signe *s*
  - Exposant *e* sur 15 bits (avec biais de 16383)
  - Mantisse *m* sur 64 bits (partie entière en bit 63, partie fractionnaire au-delà)



- Formule :

$$(-1)^s \times m \times 2^{e-16383}$$

# L'encodage des réels sous Intel 8087

- Exemple : 4001C000000000000000



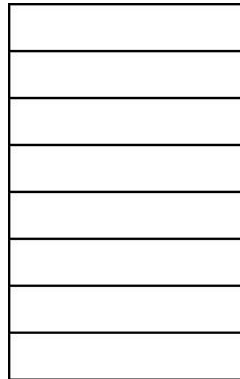
- $s = 0$
- $e = 1000000000000001_2 = 16385_{10}$
- $m = 1.100000....0_2 = 1.5_{10}$
- Formule :

$$\begin{aligned}
 (-1)^s \times m \times 2^{e-16383} &= (-1)^0 \times 1.5 \times 2^{16385-16383} \\
 &= 1 \times 1.5 \times 2^2 \\
 &= 6
 \end{aligned}$$



# Instructions du 8087 : le PUSH

- Syntaxe :



# Instructions du 8087 : le PUSH

- Syntaxe :
  - **FLD** param



# Instructions du 8087 : le PUSH

- Syntaxe :
  - **FLD** param
  - param est pushé au sommet



# Instructions du 8087 : le PUSH

- Syntaxe :

- **FLD** param
- param est pushé au sommet
- param = immédiat, mémoire, *sti*



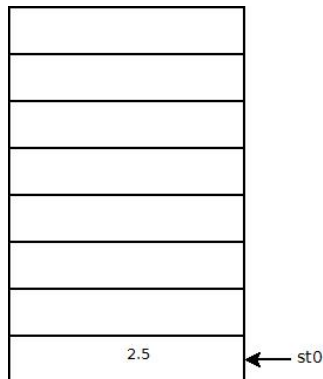
# Instructions du 8087 : le PUSH

- Syntaxe :
  - **FLD** param
  - param est pushé au sommet
  - param = immédiat, mémoire, *sti*
- Exemple :



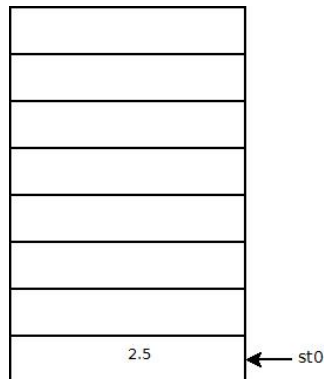
# Instructions du 8087 : le PUSH

- Syntaxe :
  - **FLD** param
  - param est pushé au sommet
  - param = immédiat, mémoire, *sti*
- Exemple :
  - **FLD tword 2.5**



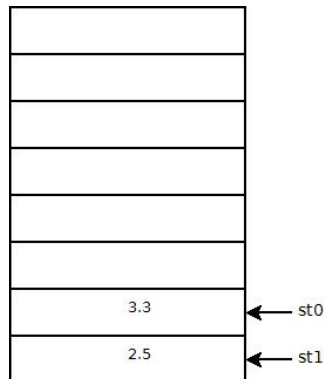
# Instructions du 8087 : le PUSH

- Syntaxe :
  - **FLD** param
  - param est poussé au sommet
  - param = immédiat, mémoire, *st*
- Exemple :
  - **FLD tword 2.5**
    - $st0 \leftarrow 2.5$



# Instructions du 8087 : le PUSH

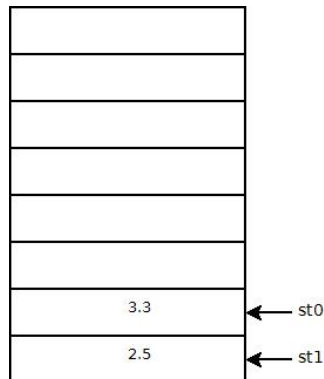
- Syntaxe :
  - **FLD** param
  - param est poussé au sommet
  - param = immédiat, mémoire, *sti*
- Exemple :
  - **FLD tword 2.5**
    - $st0 \leftarrow 2.5$
  - **FLD tword 3.3**





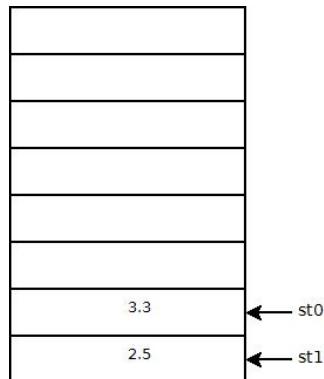
# Instructions du 8087 : le PUSH

- Syntaxe :
  - **FLD** param
  - param est pushé au sommet
  - param = immédiat, mémoire,  $st_i$
- Exemple :
  - **FLD tword 2.5**
    - $st0 \leftarrow 2.5$
  - **FLD tword 3.3**
    - $st1 \leftarrow 2.5$



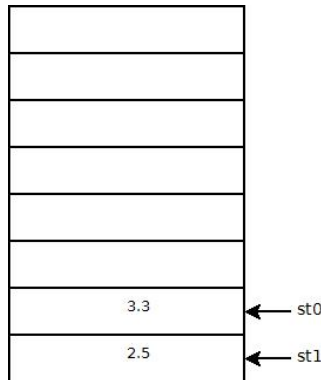
# Instructions du 8087 : le PUSH

- Syntaxe :
  - `FLD param`
  - `param` est poussé au sommet
  - `param` = immédiat, mémoire, `sti`
- Exemple :
  - `FLD tword 2.5`
    - $st0 \leftarrow 2.5$
  - `FLD tword 3.3`
    - $st1 \leftarrow 2.5$
    - $st0 \leftarrow 3.3$



# Instructions du 8087 : le PUSH

- Syntaxe :
  - **FLD** param
  - param est poussé au sommet
  - param = immédiat, mémoire,  $st_i$
- Exemple :
  - **FLD tword 2.5**
    - $st0 \leftarrow 2.5$
  - **FLD tword 3.3**
    - $st1 \leftarrow 2.5$
    - $st0 \leftarrow 3.3$
  - tword = 10 octets (*ten*)

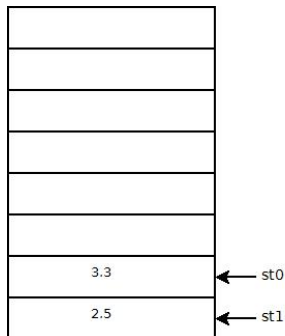


# Instructions du 8087 : le PUSH

- Variantes de FLD :
  - **FLDZ** : PUSH 0.0
  - **FLD1** : PUSH 1.0
  - **FLDPI** : PUSH  $\pi$

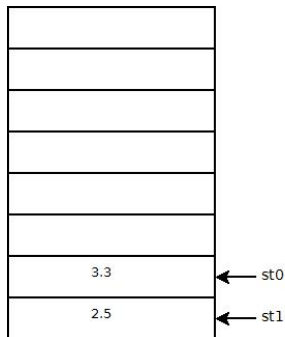
# Instructions du 8087 : le STORE

- Syntaxe : **FST** param



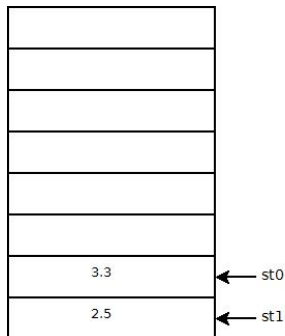
# Instructions du 8087 : le STORE

- Syntaxe : **FST** param
- Effet :



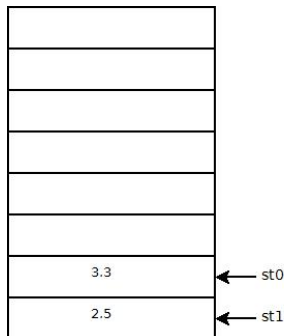
# Instructions du 8087 : le STORE

- Syntaxe : **FST** param
- Effet :
  - param  $\leftarrow$  sommet de pile



# Instructions du 8087 : le STORE

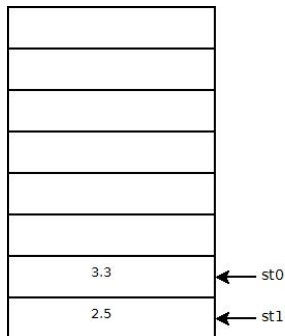
- Syntaxe : **FST** param
- Effet :
  - param  $\leftarrow$  sommet de pile
- Exemple :





# Instructions du 8087 : le STORE

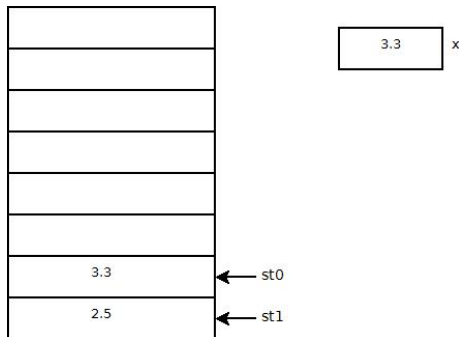
- Syntaxe : **FST** param
- Effet :
  - param  $\leftarrow$  sommet de pile
- Exemple :
  - **x DT 0.0**



0.0 x

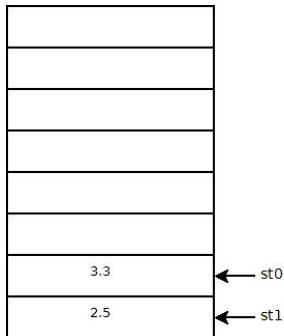
# Instructions du 8087 : le STORE

- Syntaxe : `FST param`
- Effet :
  - $\text{param} \leftarrow \text{sommet de pile}$
- Exemple :
  - `x DT 0.0`
  - `FST tword [x]`



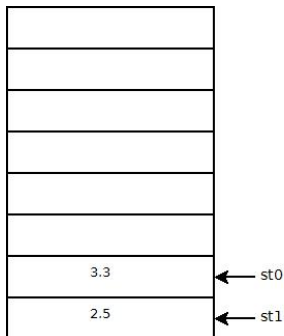
# Instructions du 8087 : le POP

- Syntaxe : **FSTP** param



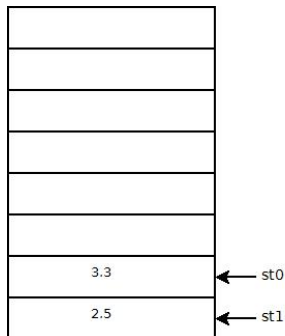
# Instructions du 8087 : le POP

- Syntaxe : **FSTP** param
- Effet :



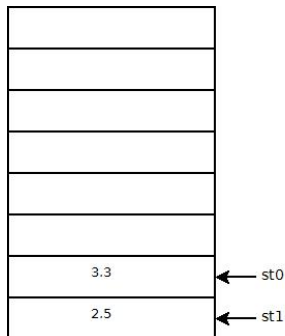
# Instructions du 8087 : le POP

- Syntaxe : **FSTP** param
- Effet :
  - $\text{param} \leftarrow \text{sommet de pile}$



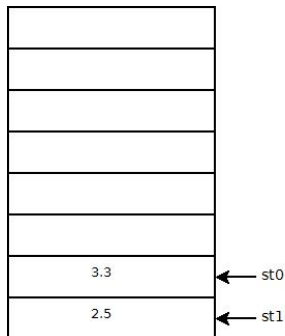
# Instructions du 8087 : le POP

- Syntaxe : **FSTP** param
- Effet :
  - $\text{param} \leftarrow \text{sommet de pile}$
  - POP



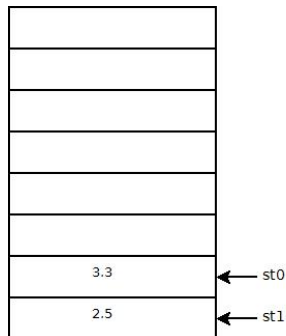
# Instructions du 8087 : le POP

- Syntaxe : **FSTP** param
- Effet :
  - $\text{param} \leftarrow \text{sommet de pile}$
  - POP
- Exemple :



# Instructions du 8087 : le POP

- Syntaxe : **FSTP** param
- Effet :
  - $\text{param} \leftarrow \text{sommet de pile}$
  - POP
- Exemple :
  - **x DT 0.0**

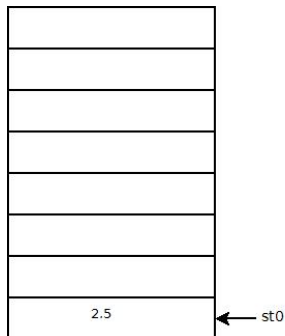


0.0 x



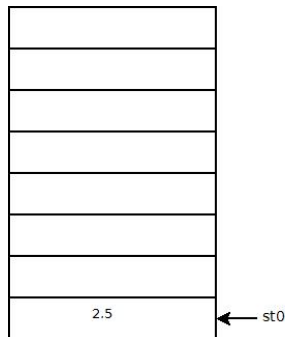
# Instructions du 8087 : le POP

- Syntaxe : `FSTP param`
- Effet :
  - $\text{param} \leftarrow \text{sommet de pile}$
  - POP
- Exemple :
  - `x DT 0.0`
  - `FSTP tword [x]`



# Instructions du 8087 : le POP

- Syntaxe : **FSTP** param
- Effet :
  - $\text{param} \leftarrow \text{sommet de pile}$
  - POP
- Exemple :
  - **x DT 0.0**
  - **FSTP tword [x]**
- **P** = « *pop* »



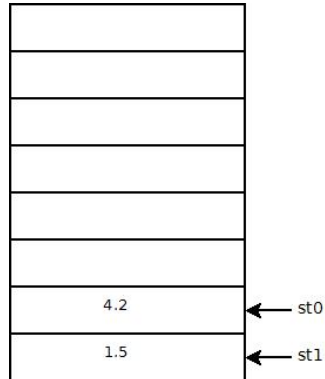
3.3 x

# Opérations arithmétiques du 8087

- Opérations binaires
  - Addition
  - Soustraction
  - Multiplication
  - Division
- Opérations unaires
  - Racine carrée
  - Valeur absolue
  - Changement de signe
  - Conversion réel-entier
  - Cosinus
  - ...

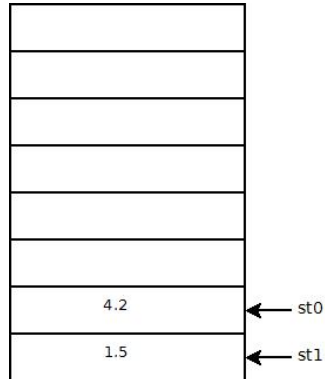
# Addition

- `FADD sti, stj`



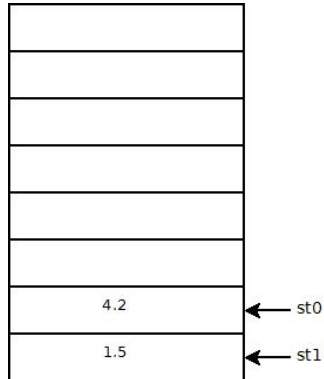
# Addition

- `FADD sti, stj`
  - $sti \leftarrow sti + stj$



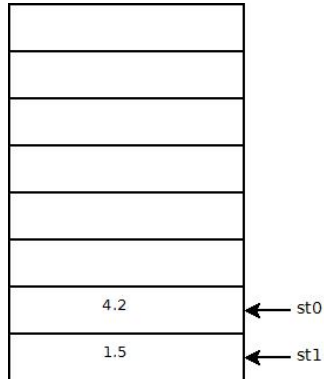
# Addition

- `FADD sti, stj`
  - $sti \leftarrow sti + stj$
- Exemple :



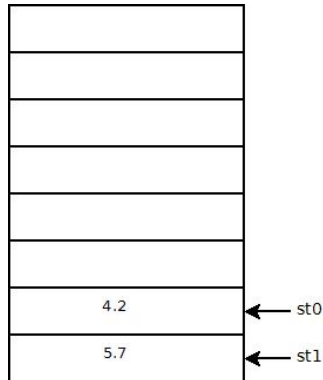
# Addition

- `FADD sti, stj`
  - $sti \leftarrow sti + stj$
- Exemple :
  - `FADD st1, st0`



# Addition

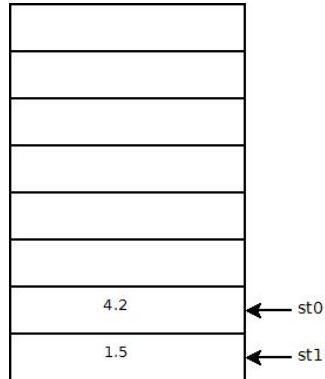
- `FADD sti, stj`
  - $sti \leftarrow sti + stj$
- Exemple :
  - `FADD st1, st0`
  - $st1 \leftarrow st1 + st0$





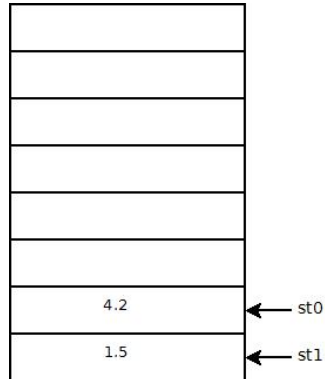
# Addition avec pop

- `FADDP sti, stj`



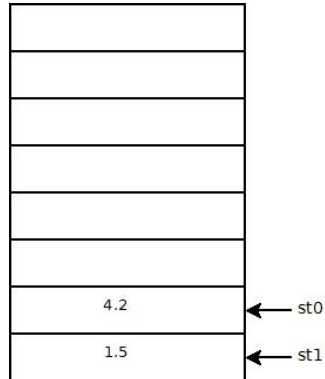
# Addition avec pop

- `FADDP sti, stj`
  - $sti \leftarrow sti + stj$



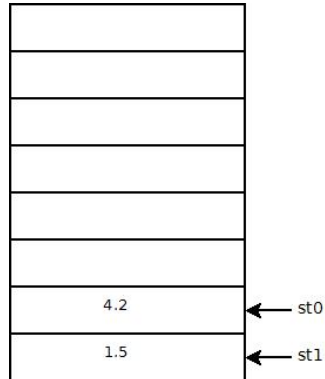
# Addition avec pop

- `FADDP sti, stj`
  - $sti \leftarrow sti + stj$
  - POP



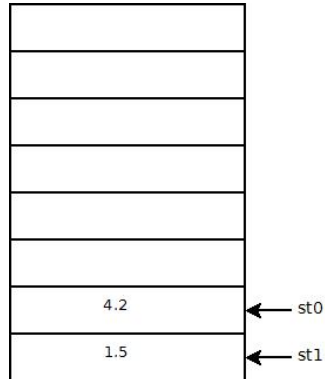
# Addition avec pop

- `FADDP sti, stj`
  - $sti \leftarrow sti + stj$
  - POP
  - `P = « pop »`



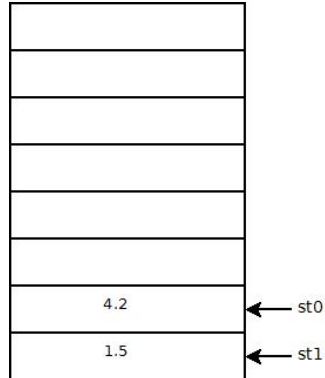
# Addition avec pop

- `FADDP sti, stj`
  - $sti \leftarrow sti + stj$
  - POP
  - `P` = « *pop* »
- Exemple :



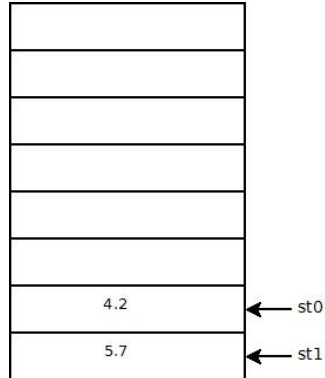
# Addition avec pop

- `FADDP sti, stj`
  - $sti \leftarrow sti + stj$
  - POP
  - `P` = « *pop* »
- Exemple :
  - `FADDP st1, st0`



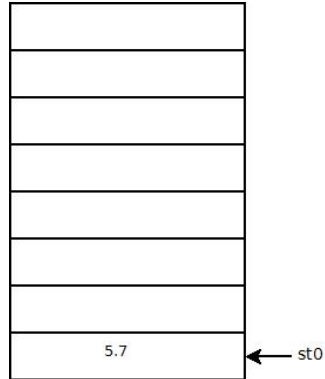
# Addition avec pop

- `FADDP sti, stj`
  - $sti \leftarrow sti + stj$
  - POP
  - `P` = « *pop* »
- Exemple :
  - `FADDP st1, st0`
  - $st1 \leftarrow st1 + st0$



# Addition avec pop

- `FADDP sti, stj`
  - $sti \leftarrow sti + stj$
  - POP
  - `P` = « *pop* »
- Exemple :
  - `FADDP st1, st0`
  - $st1 \leftarrow st1 + st0$
  - POP





# Addition : raccourcis

- `FADDP sti`

# Addition : raccourcis

- `FADDP sti`
  - raccourci pour `FADDP sti, st0`

# Addition : raccourcis

- `FADDP sti`
  - raccourci pour `FADDP sti, st0`
- `FADDP`

## Addition : raccourcis

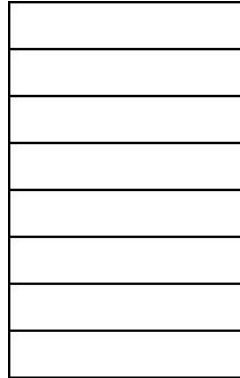
- `FADDP sti`
  - raccourci pour `FADDP sti, st0`
- `FADDP`
  - raccourci pour `FADDP st1, st0`

# Addition : exemple 1

- $2.5 + 3.3$

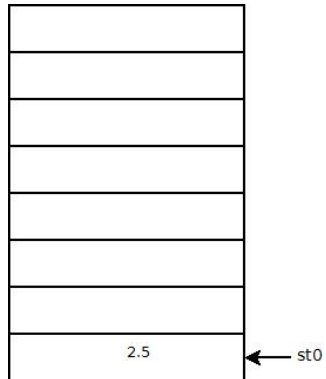

## Addition : exemple 1

- $2.5 + 3.3$
- `FLD tword 2.5`



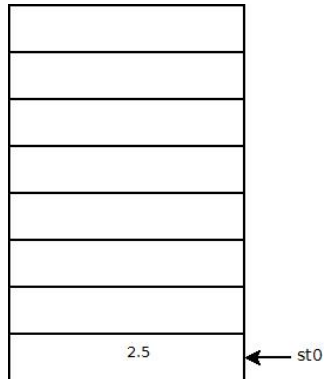
## Addition : exemple 1

- $2.5 + 3.3$
- `FLD tword 2.5`



## Addition : exemple 1

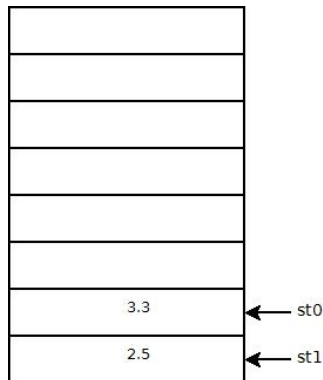
- $2.5 + 3.3$
- `FLD tword 2.5`
- `FLD tword 3.3`





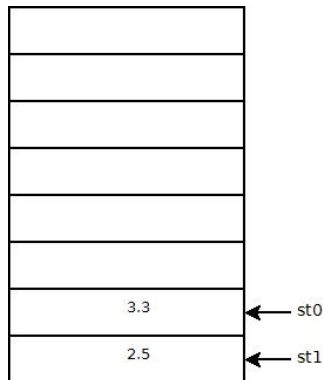
## Addition : exemple 1

- $2.5 + 3.3$
- `FLD tword 2.5`
- `FLD tword 3.3`



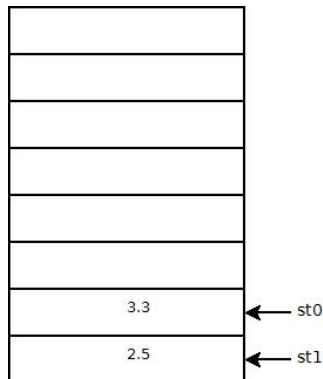
## Addition : exemple 1

- $2.5 + 3.3$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP st1, st0`



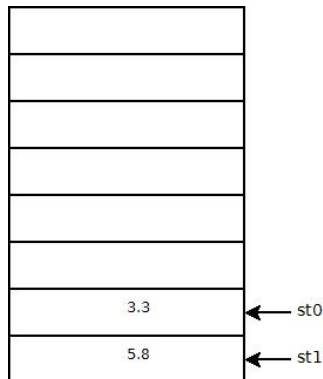
## Addition : exemple 1

- $2.5 + 3.3$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP st1, st0`
  - $st1 \leftarrow st1 + st0$



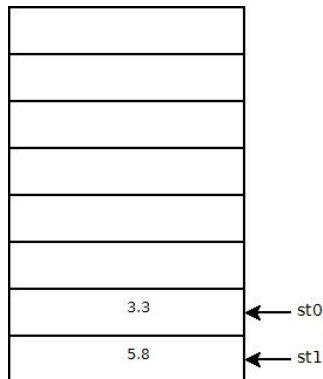
## Addition : exemple 1

- $2.5 + 3.3$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP st1, st0`
  - $st1 \leftarrow st1 + st0$



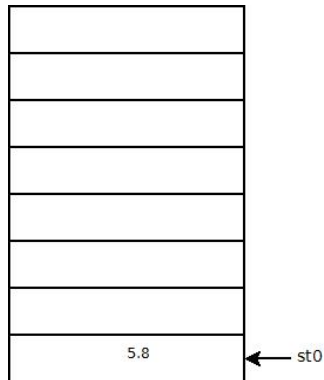
## Addition : exemple 1

- $2.5 + 3.3$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP st1, st0`
  - $st1 \leftarrow st1 + st0$
  - `POP`



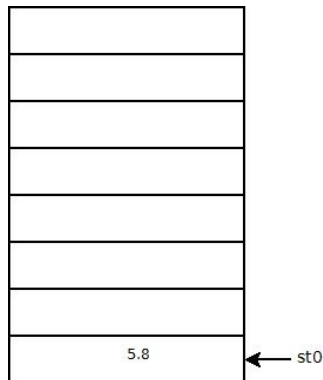
## Addition : exemple 1

- $2.5 + 3.3$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP st1, st0`
  - $st1 \leftarrow st1 + st0$
  - `POP`



## Addition : exemple 1

- $2.5 + 3.3$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP st1, st0`
  - $st1 \leftarrow st1 + st0$
  - `POP`
- **Note** : on abrègera désormais  
`FADDP st1, st0` en `FADDP`



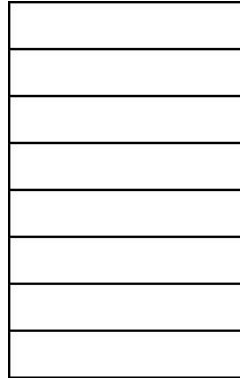
## Addition : exemple 2

- $2.5 + 3.3 + 4.1$



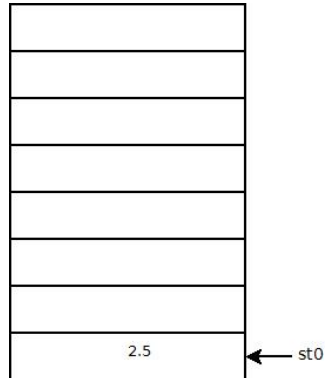

## Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`



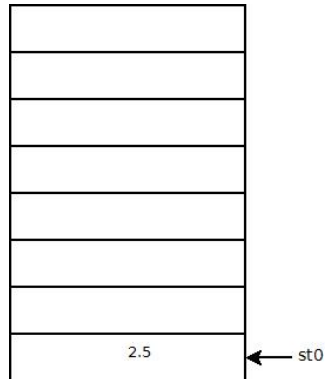
## Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`



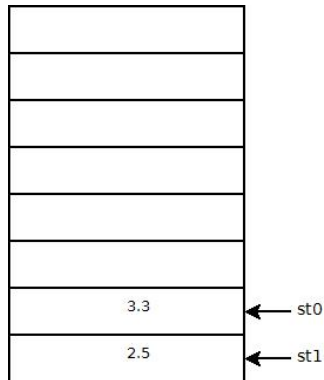
## Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`
- `FLD tword 3.3`



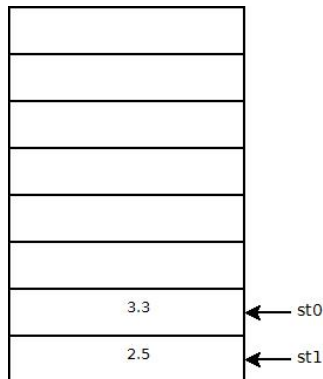
## Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`
- `FLD tword 3.3`



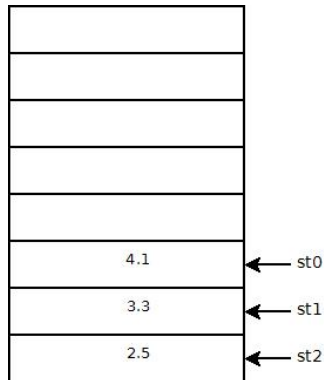
## Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FLD tword 4.1`



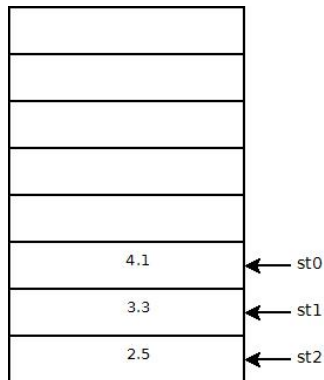
## Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FLD tword 4.1`



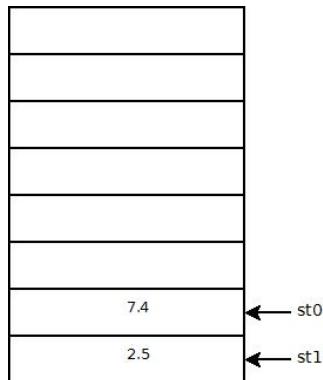
## Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FLD tword 4.1`
- `FADDP`



## Addition : exemple 2

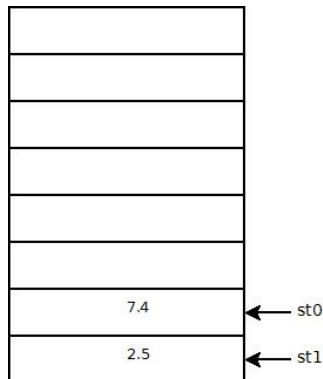
- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FLD tword 4.1`
- `FADDP`





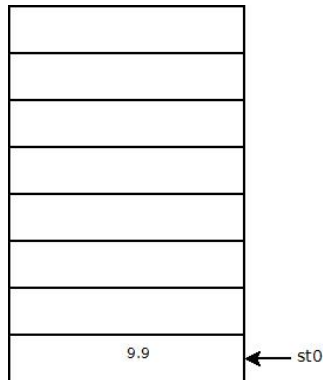
## Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FLD tword 4.1`
- `FADDP`
- `FADDP`



## Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FLD tword 4.1`
- `FADDP`
- `FADDP`



# Autres opérations binaires

- Soustraction
  - FSUB/FSUBP
- Multiplication
  - FMUL/FMULP
- Division
  - FDIV/FDIVP
- Même syntaxe que l'addition (raccourcis inclus)

# Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$

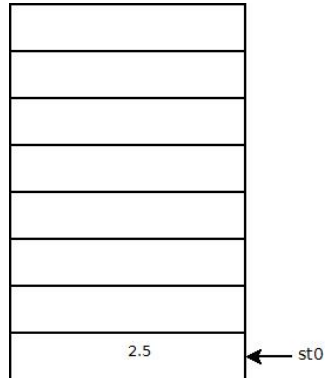

# Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`



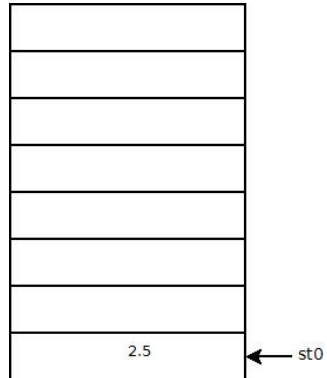
## Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`



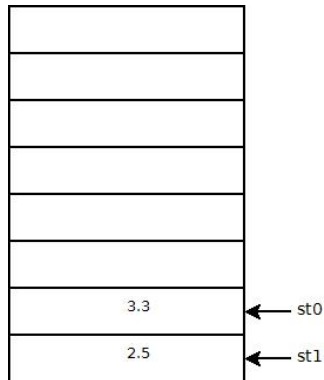
# Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`



## Exemple

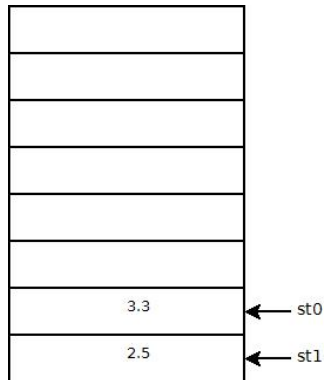
- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`





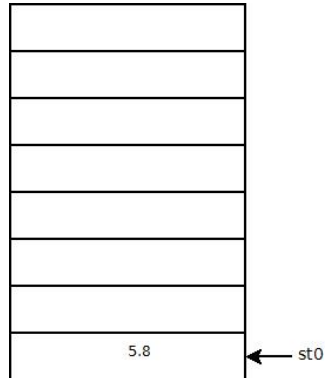
## Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`



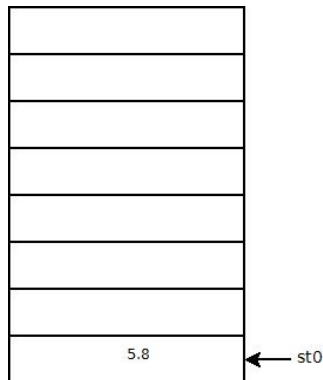
## Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`



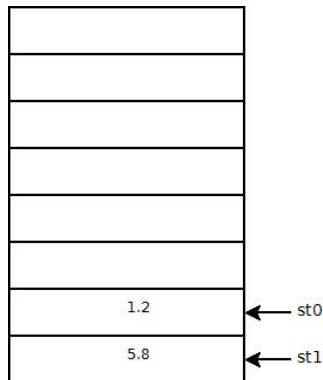
## Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`
- `FLD tword 1.2`



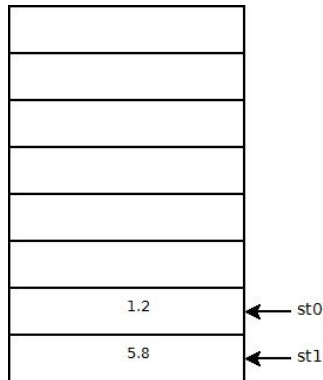
## Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`
- `FLD tword 1.2`



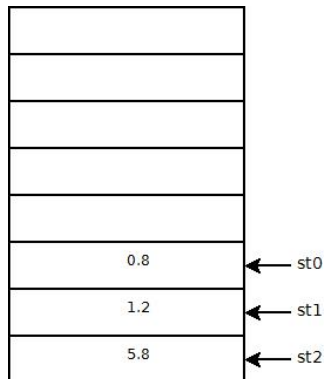
## Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`
- `FLD tword 1.2`
- `FLD tword 0.8`



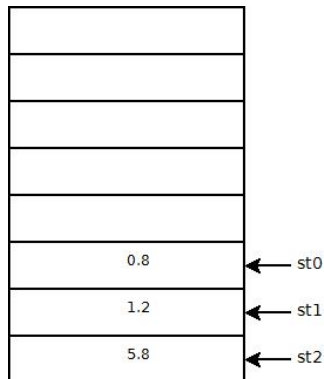
## Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`
- `FLD tword 1.2`
- `FLD tword 0.8`



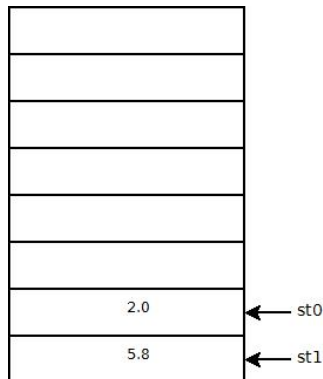
## Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`
- `FLD tword 1.2`
- `FLD tword 0.8`
- `FADDP`



## Exemple

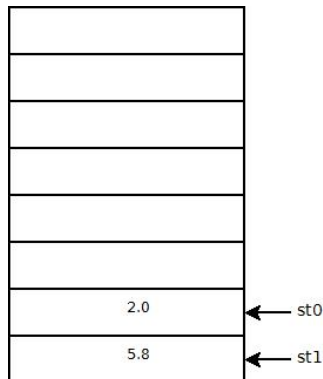
- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`
- `FLD tword 1.2`
- `FLD tword 0.8`
- `FADDP`





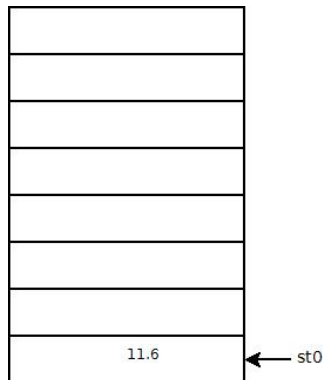
# Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`
- `FLD tword 1.2`
- `FLD tword 0.8`
- `FADDP`
- `FMULP`



## Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `FLD tword 2.5`
- `FLD tword 3.3`
- `FADDP`
- `FLD tword 1.2`
- `FLD tword 0.8`
- `FADDP`
- `FMULP`



# Opérations unaires

- valeur absolue

# Opérations unaires

- valeur absolue
  - **FABS**

# Opérations unaires

- valeur absolue
  - **FABS**
  - $st0 \leftarrow |st0|$

# Opérations unaires

- valeur absolue
  - **FABS**
  - $st0 \leftarrow |st0|$
- Racine carrée

# Opérations unaires

- valeur absolue
  - **FABS**
  - $st0 \leftarrow |st0|$
- Racine carrée
  - **FSQRT**

# Opérations unaires

- valeur absolue
  - **FABS**
  - $st0 \leftarrow |st0|$
- Racine carrée
  - **FSQRT**
  - $st0 \leftarrow \sqrt{st0}$



# Opérations unaires

- valeur absolue
  - **FABS**
  - $st0 \leftarrow |st0|$
- Racine carrée
  - **FSQRT**
  - $st0 \leftarrow \sqrt{st0}$
- Cosinus

# Opérations unaires

- valeur absolue
  - **FABS**
  - $st0 \leftarrow |st0|$
- Racine carrée
  - **FSQRT**
  - $st0 \leftarrow \sqrt{st0}$
- Cosinus
  - **FCOS**

# Opérations unaires

- valeur absolue
  - **FABS**
  - $st0 \leftarrow |st0|$
- Racine carrée
  - **FSQRT**
  - $st0 \leftarrow \sqrt{st0}$
- Cosinus
  - **FCOS**
  - $st0 \leftarrow \cos(st0)$

# Opérations unaires

- valeur absolue
  - **FABS**
  - $st0 \leftarrow |st0|$
- Racine carrée
  - **FSQRT**
  - $st0 \leftarrow \sqrt{st0}$
- Cosinus
  - **FCOS**
  - $st0 \leftarrow \cos(st0)$
- Etc...

# Conversions réel $\longleftrightarrow$ entier

- Rajout d'un **I** après le **F** initial

# Conversions réel $\iff$ entier

- Rajout d'un **I** après le **F** initial
- Conversion automatique réel  $\iff$  entier

# Conversions réel $\Longleftrightarrow$ entier

- Rajout d'un **I** après le **F** initial
- Conversion automatique réel  $\Longleftrightarrow$  entier
- Ex : **FILD** **eax**

# Conversions réel $\Longleftrightarrow$ entier

- Rajout d'un **I** après le **F** initial
- Conversion automatique réel  $\Longleftrightarrow$  entier
- Ex : **FILD** **eax**
  - **eax** (entier 32 bits) converti vers réel 80 bits puis pushé



# Conversions réel $\longleftrightarrow$ entier

- Rajout d'un **I** après le **F** initial
- Conversion automatique réel  $\longleftrightarrow$  entier
- Ex : **FILD** *eax*
  - *eax* (entier 32 bits) converti vers réel 80 bits puis pushé
- Ex : **FISTP** *eax*

# Conversions réel $\iff$ entier

- Rajout d'un **I** après le **F** initial
- Conversion automatique réel  $\iff$  entier
- Ex : **FILD** **eax**
  - **eax** (entier 32 bits) converti vers réel 80 bits puis pushé
- Ex : **FISTP** **eax**
  - réel 80 bits converti vers entier 32 bits puis poppé vers **eax**

# Exemple complet

```
global main
section .data
x1      DT    3.5      ; point1
y1      DT    1.6
x2      DT    25.2     ; point2
y2      DT    31.3

section .bss
dist    RESD 1  ; distance entre les points (convertie en entier)

section .text
main:
    FINIT          ; initialiser la pile
    FLD            tword [x1]
    FLD            tword [x2]
    FSUBP
    FLD st0

    FLD            tword [y1]
    FLD            tword [y2]
    FSUBP
    FLD st0

    FADDP
    FSQRT
    FISTP dword [dist]

    MOV            eax,1
    MOV            ebx,0
    INT            0x80
```

# Références

- Manuel Intel (chap. 8)
  - <http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-1-manual.pdf>