Microprocesseurs (MIC)

Chap. 7 : Le co-processeur Intel 8087

Sommaire

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 représentant des nombres à virgule fixe implicite
  - Ex : $2.5 + 3.4$
    - On représente les nombre comme 250 et 340 (si deux décimales implicites)
    - La somme donne 590 ce qui correspond à 5.9
- Le 8087 introduit des instructions de calcul de nombres « à virgule flottante »

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»

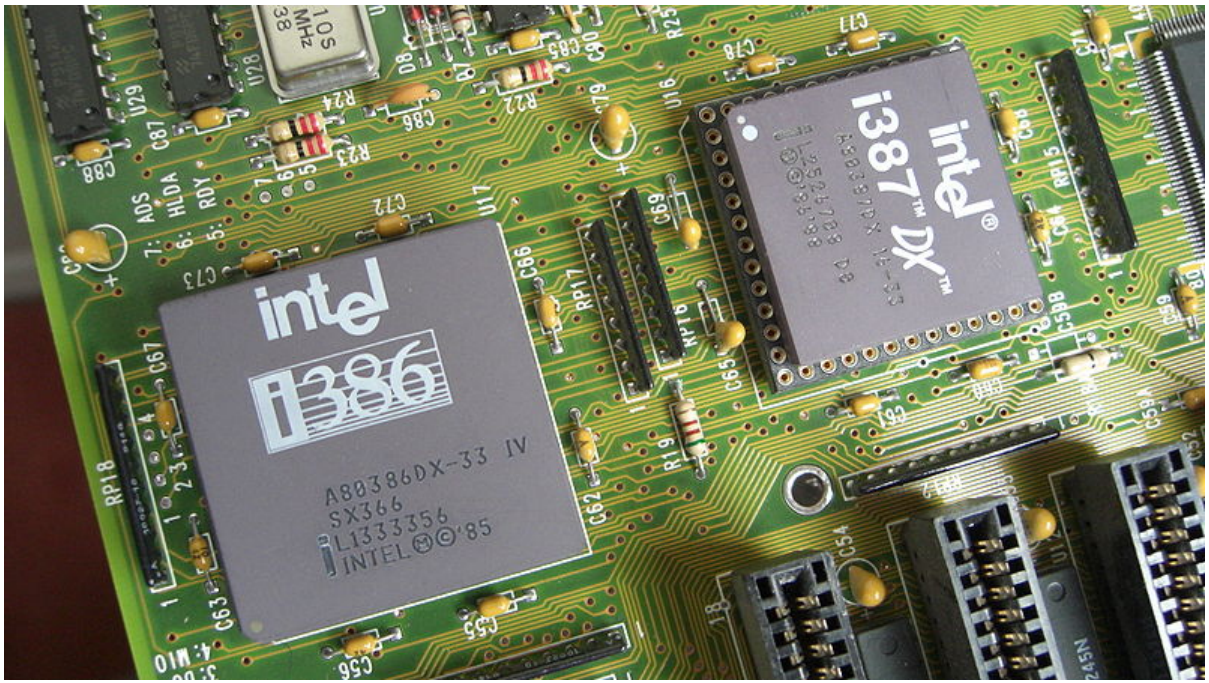
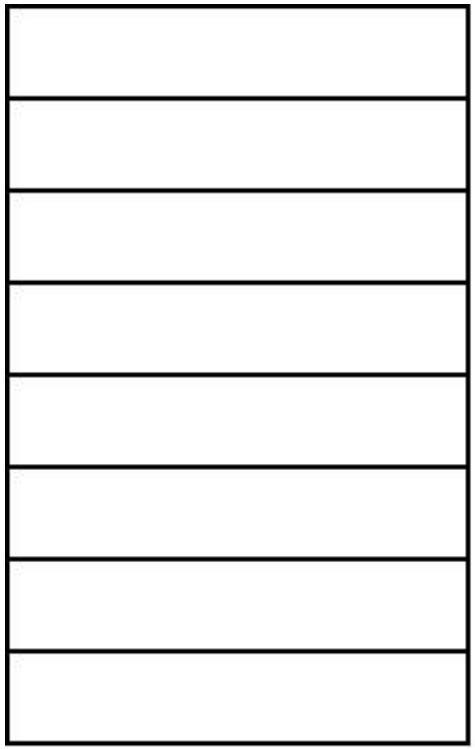


Figure: Processeur 80386 avec co-processeur 80387 séparé

Sommaire

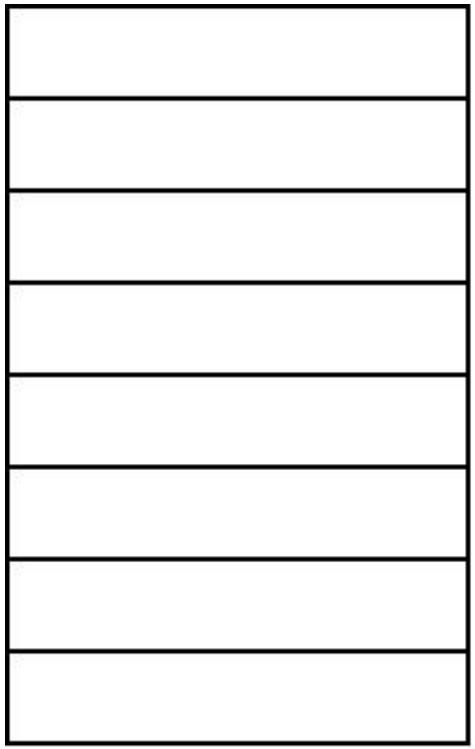
Le calcul par pile

- Calcul par pile



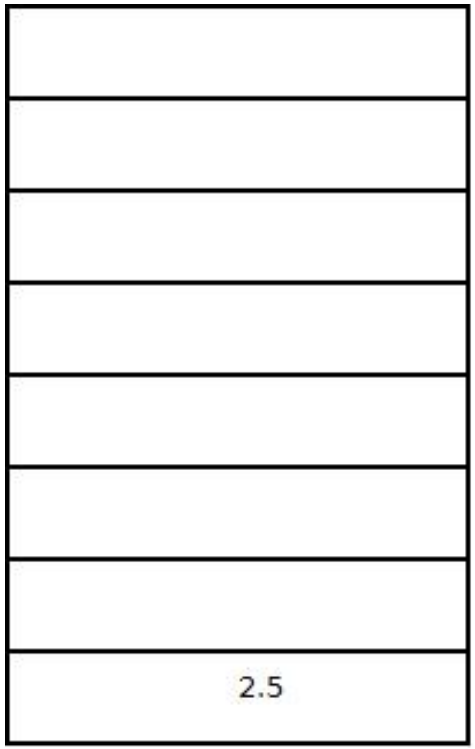
Le calcul par pile

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



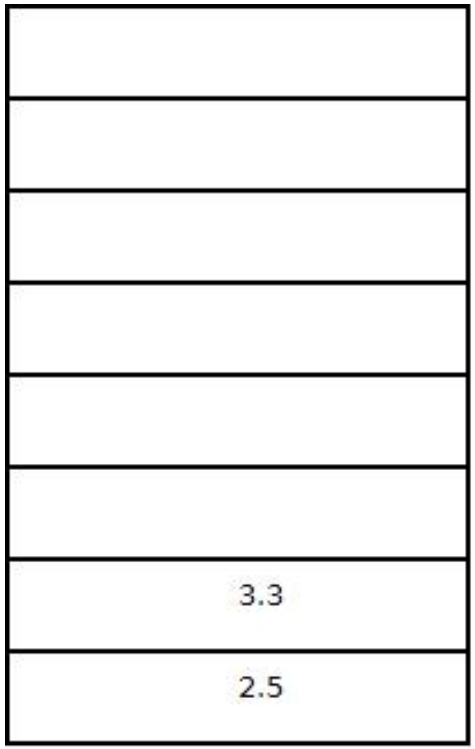
Le calcul par pile

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



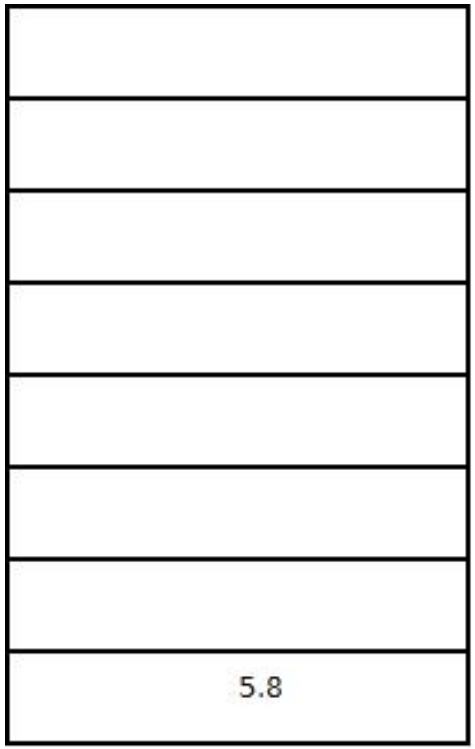
Le calcul par pile

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



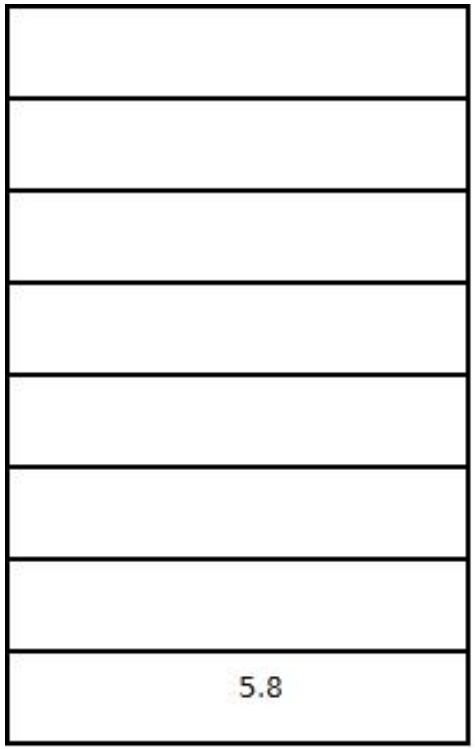
Le calcul par pile

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



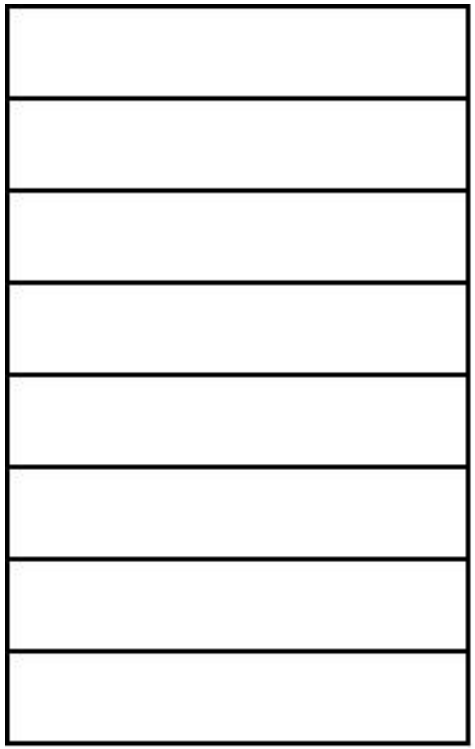
Le calcul par pile

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



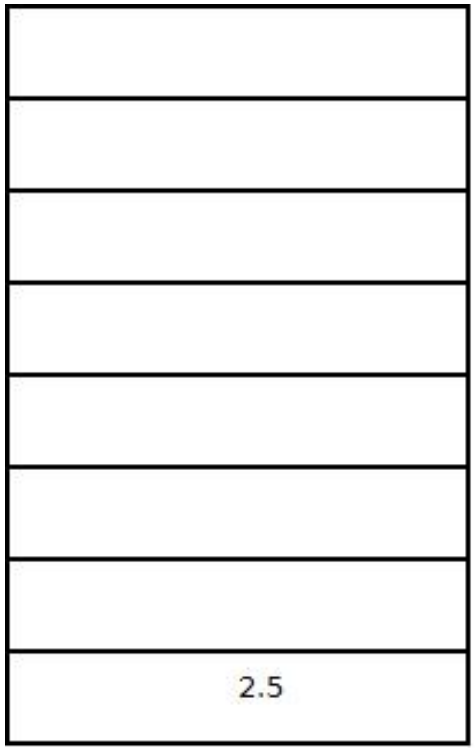
Le calcul par pile

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



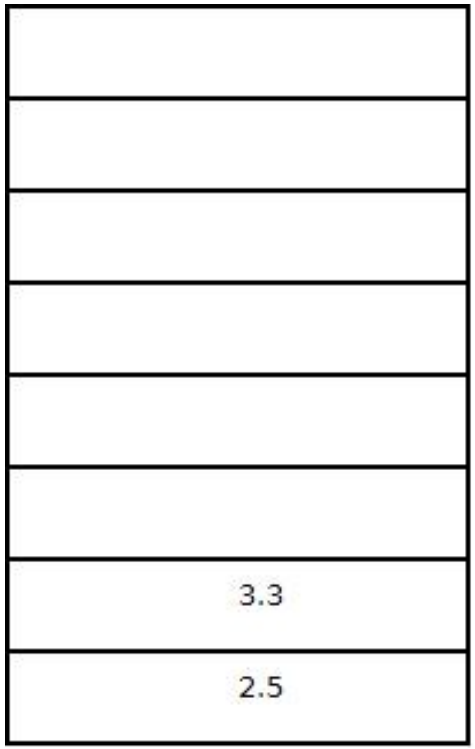
Le calcul par pile

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



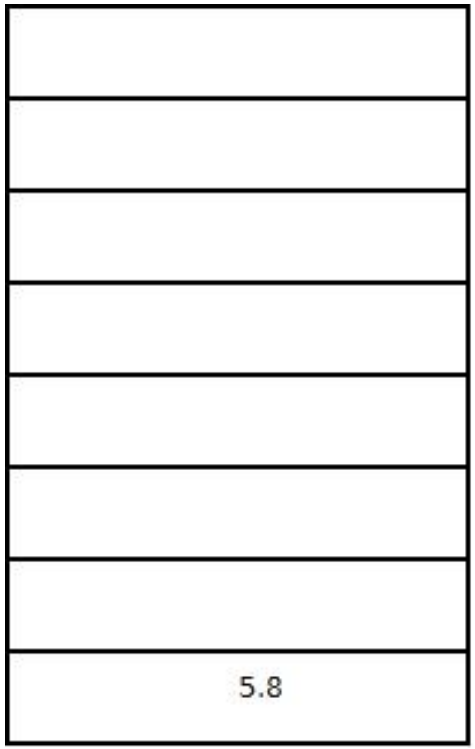
Le calcul par pile

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



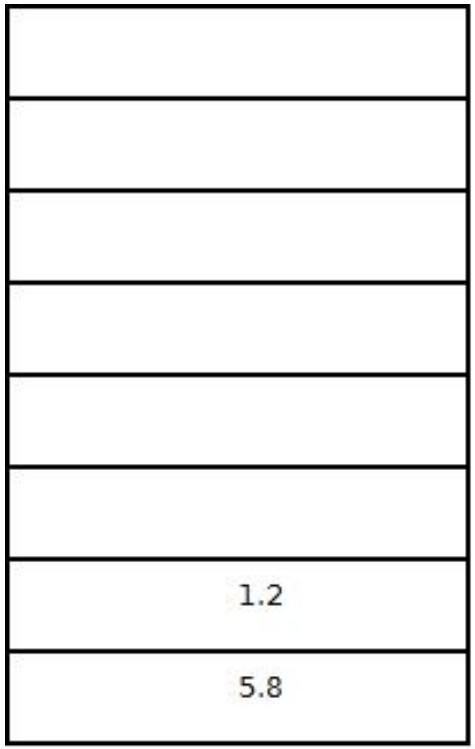
Le calcul par pile

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



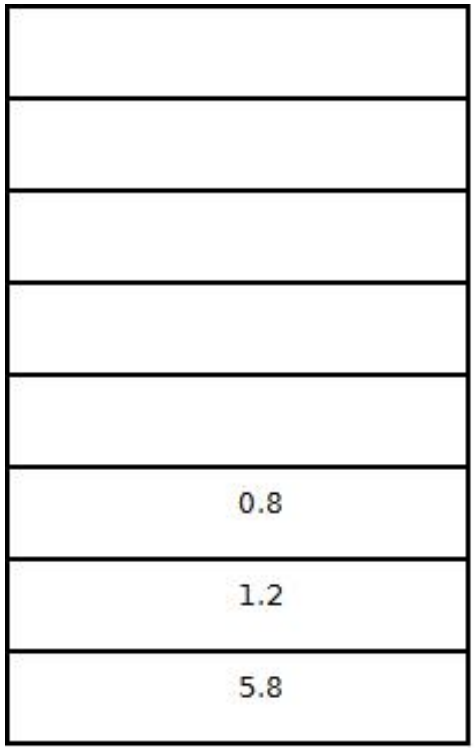
Le calcul par pile

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



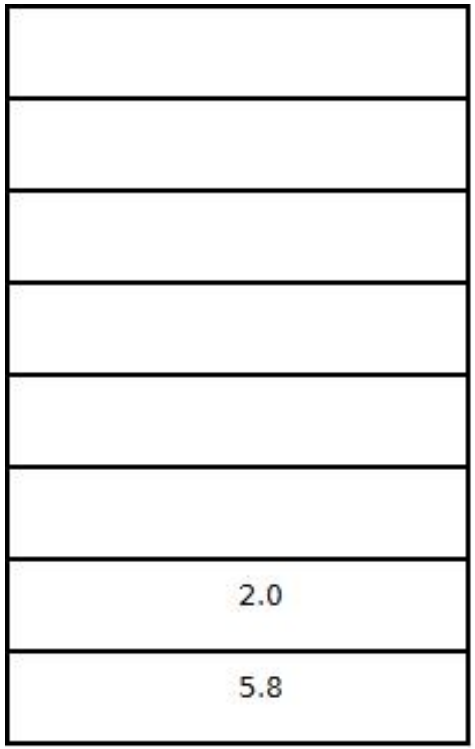
Le calcul par pile

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



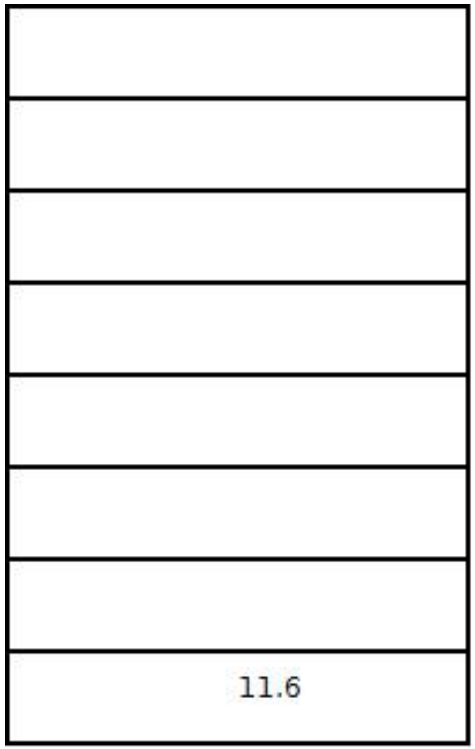
Le calcul par pile

- 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



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



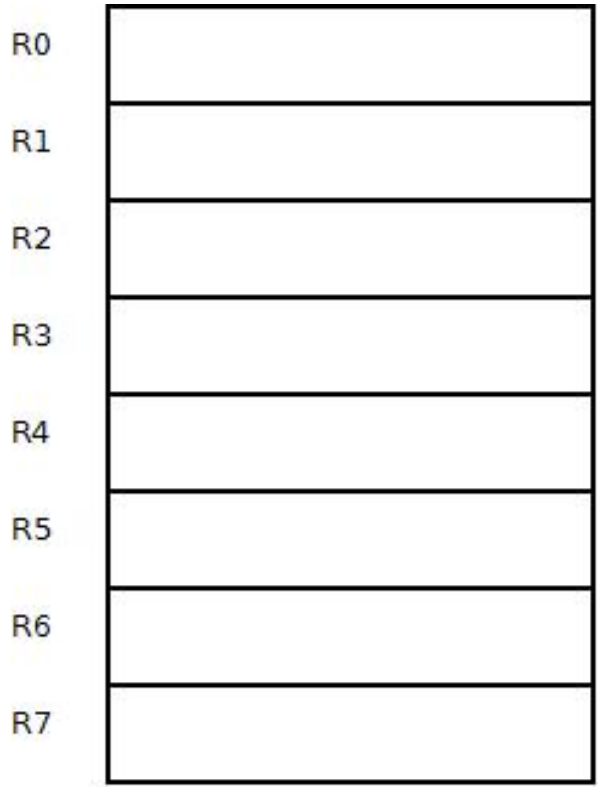
Le calcul par pile

- 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

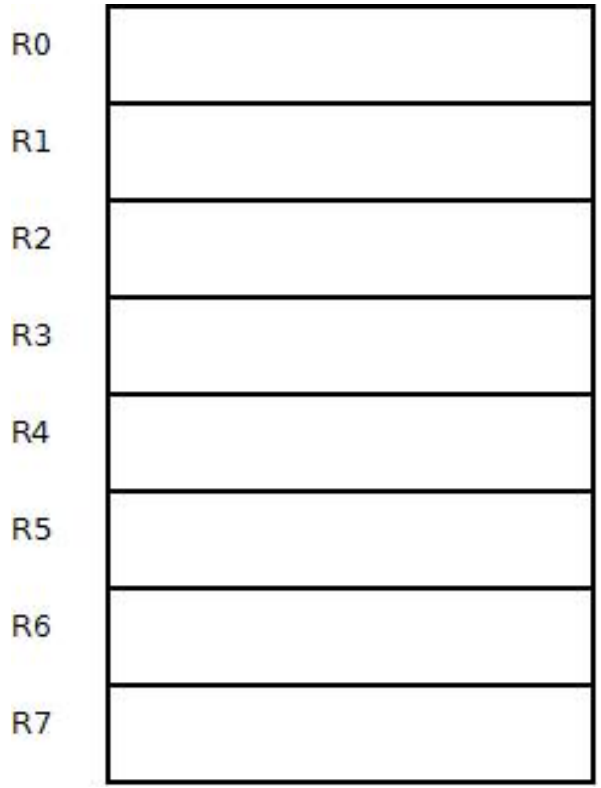
La pile du 8087 : registres

- Pile hardware dans le co-processeur



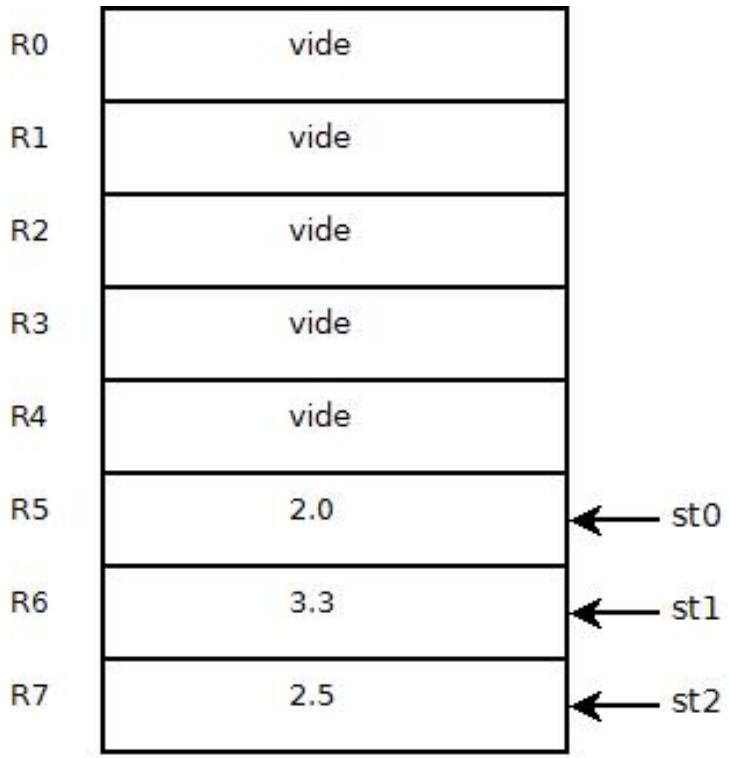
La pile du 8087 : registres

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



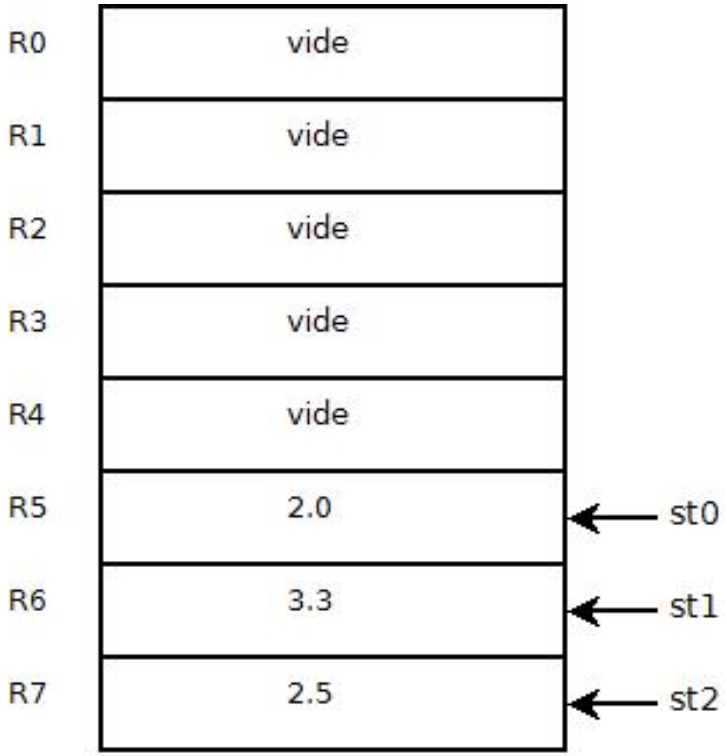
La pile du 8087 : utilisation

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



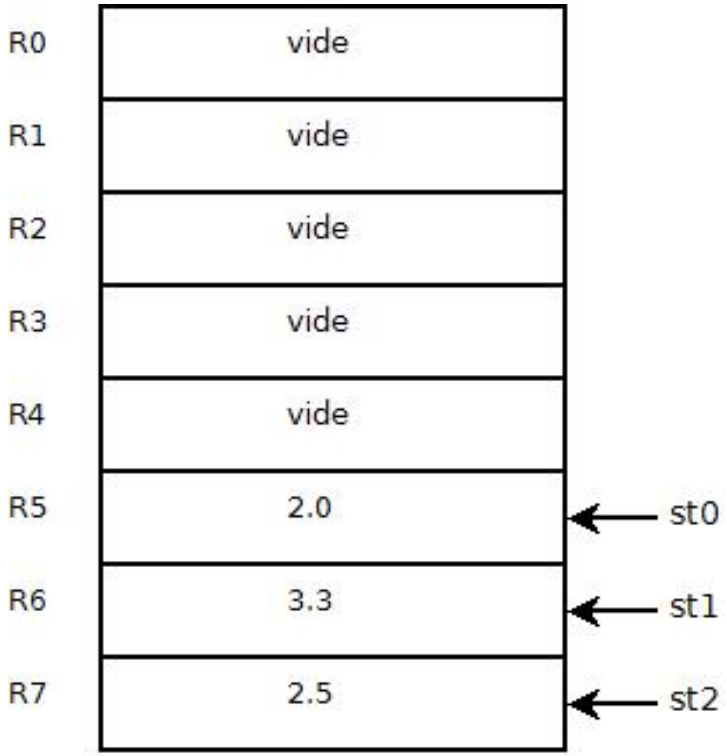
La pile du 8087 : utilisation

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



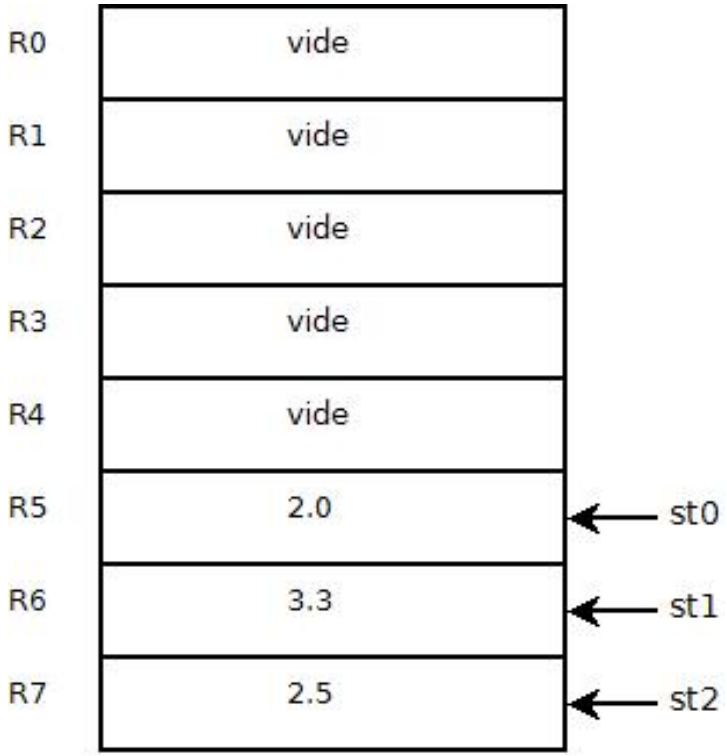
La pile du 8087 : utilisation

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



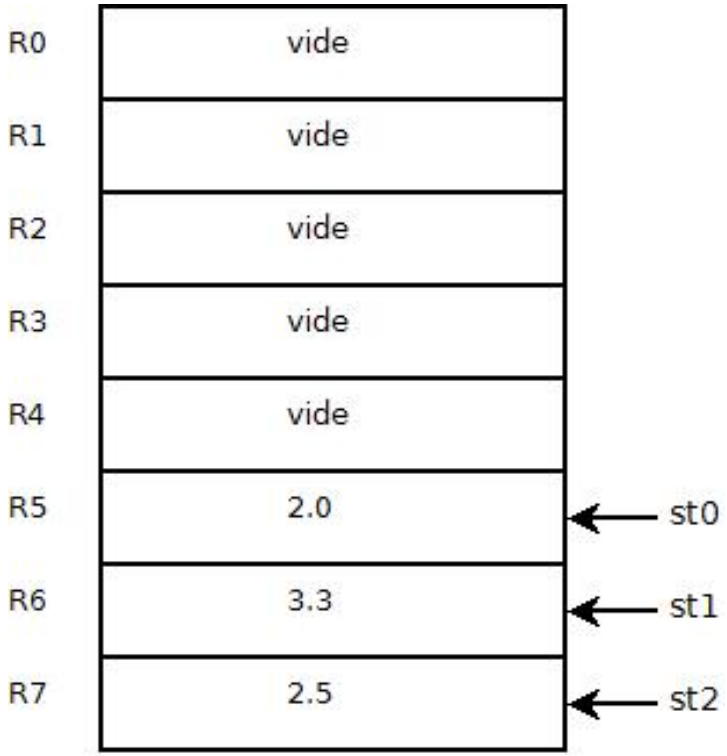
La pile du 8087 : utilisation

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



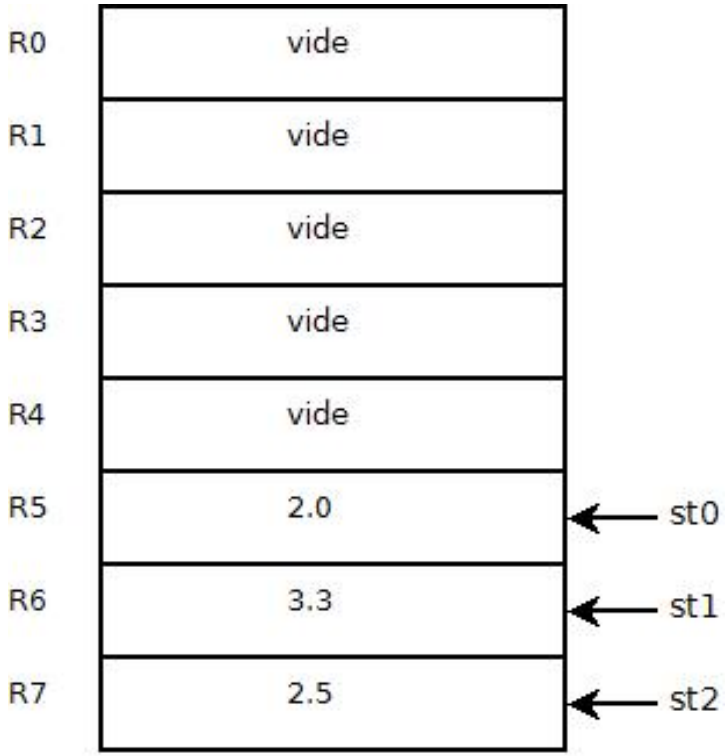
La pile du 8087 : utilisation

- 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



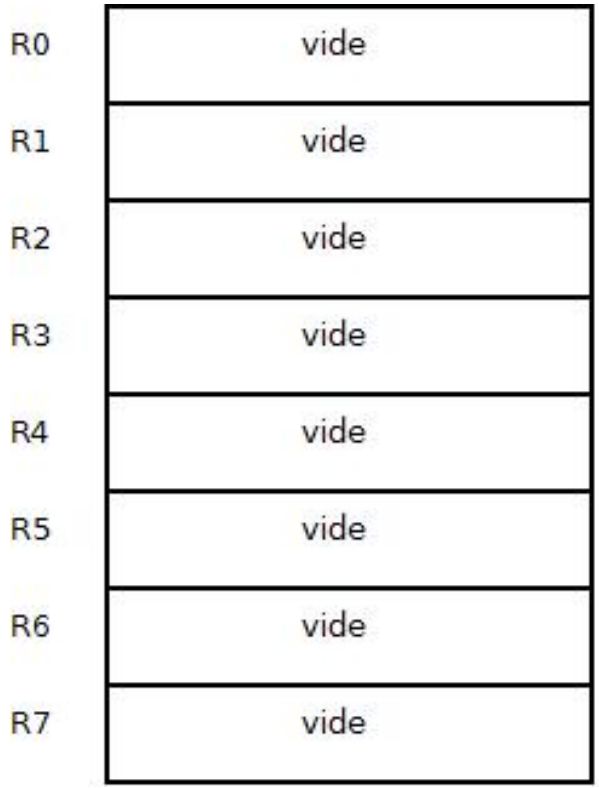
La pile du 8087 : utilisation

- 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
  - ...



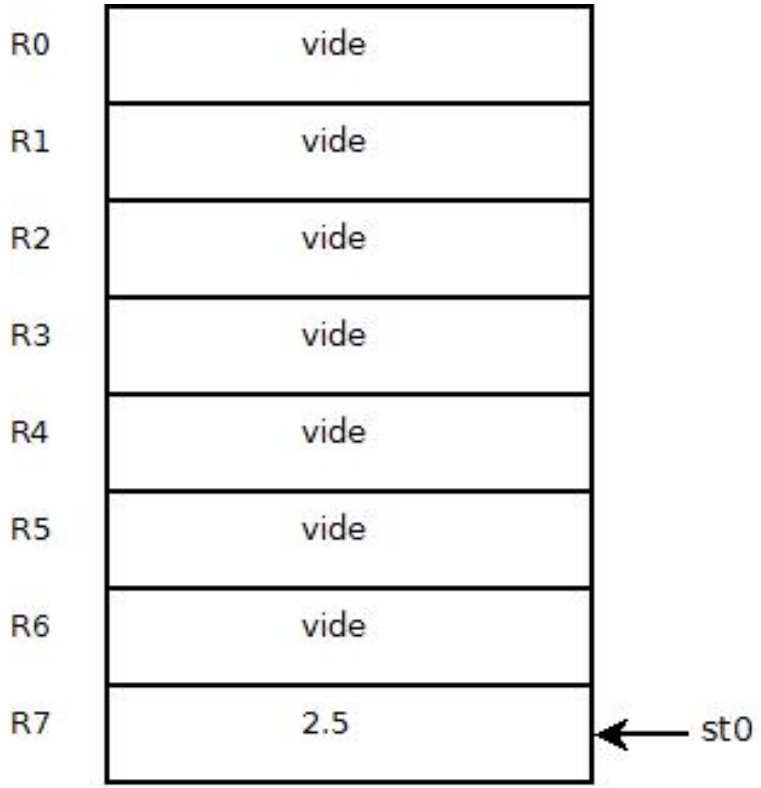
La pile du 8087 : exemple 1

- Pile vide au départ



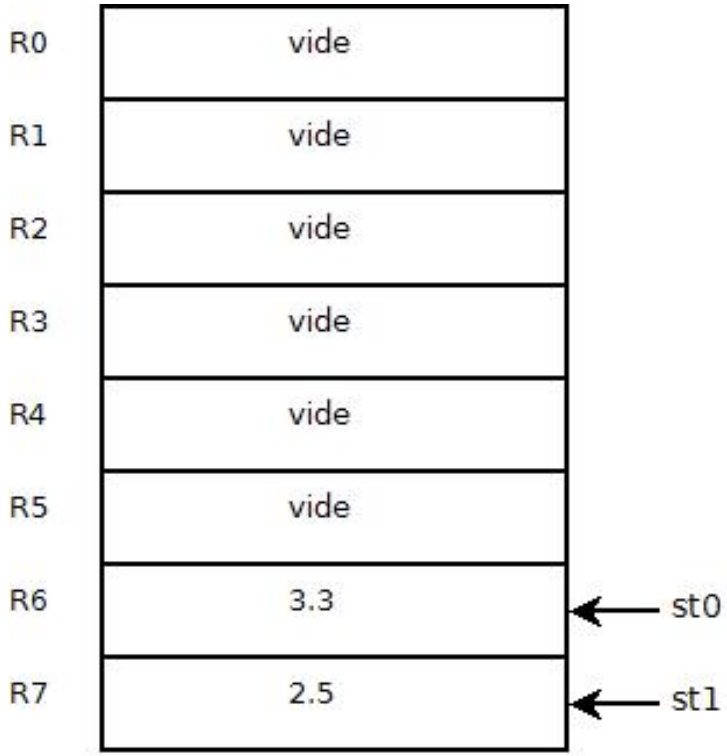
La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5



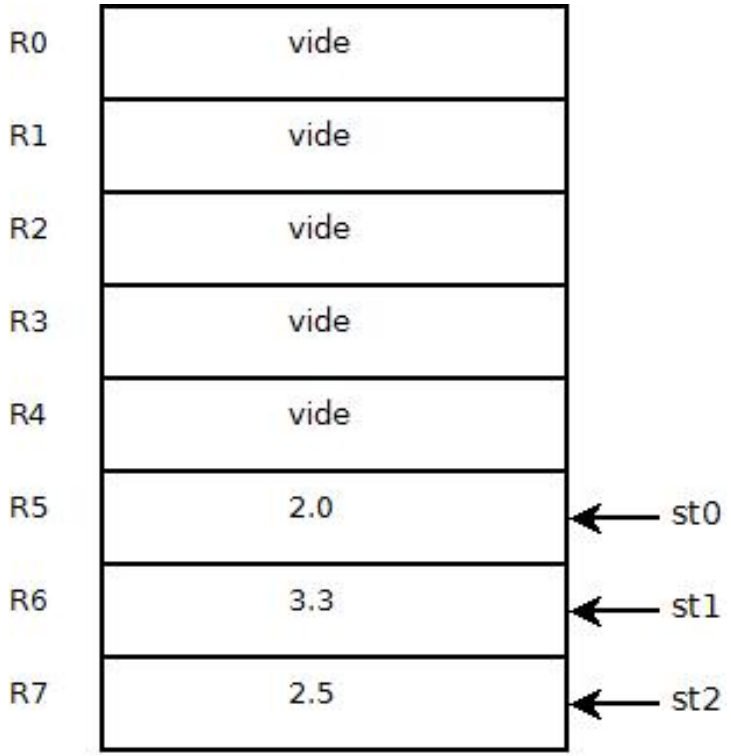
La pile du 8087 : exemple 1

- Pile vide au départ
- PUSH 2.5
- PUSH 3.3



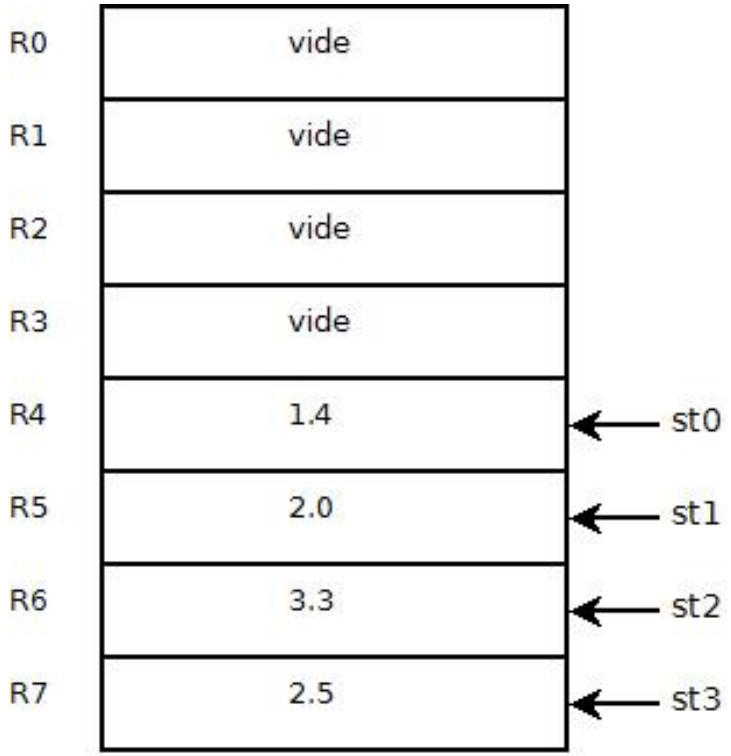
La pile du 8087 : exemple 1

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



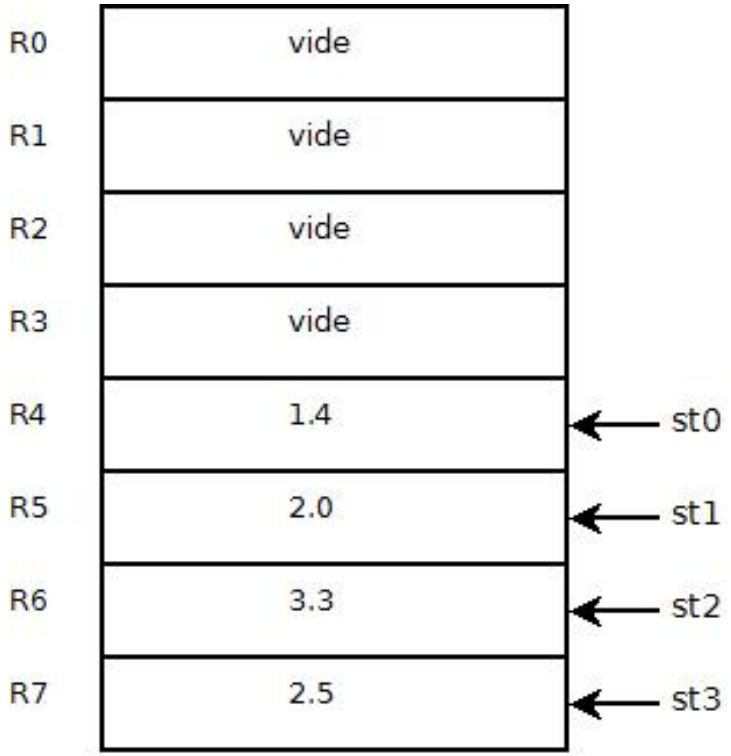
La pile du 8087 : exemple 1

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



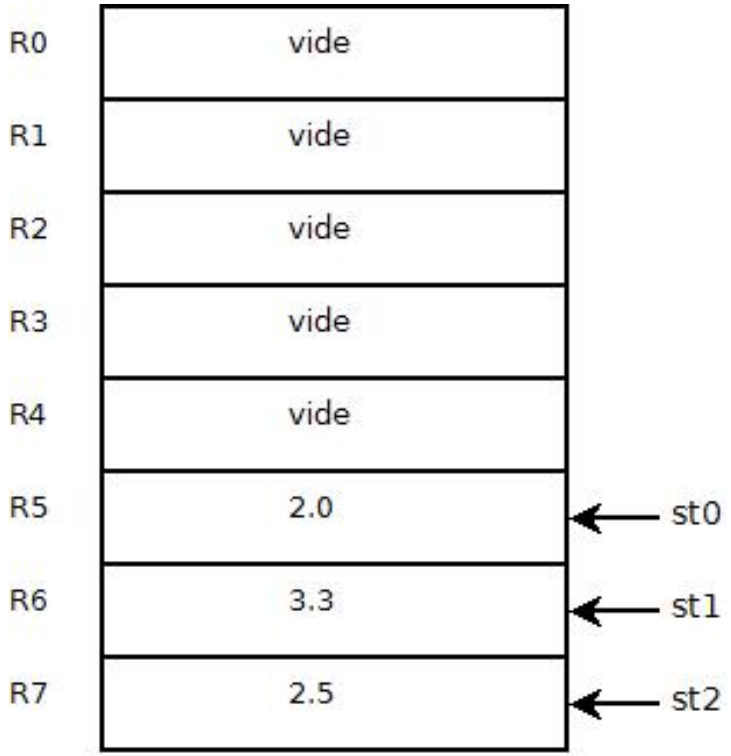
La pile du 8087 : exemple 2

- Partons de la pile obtenue dans l'exemple précédent



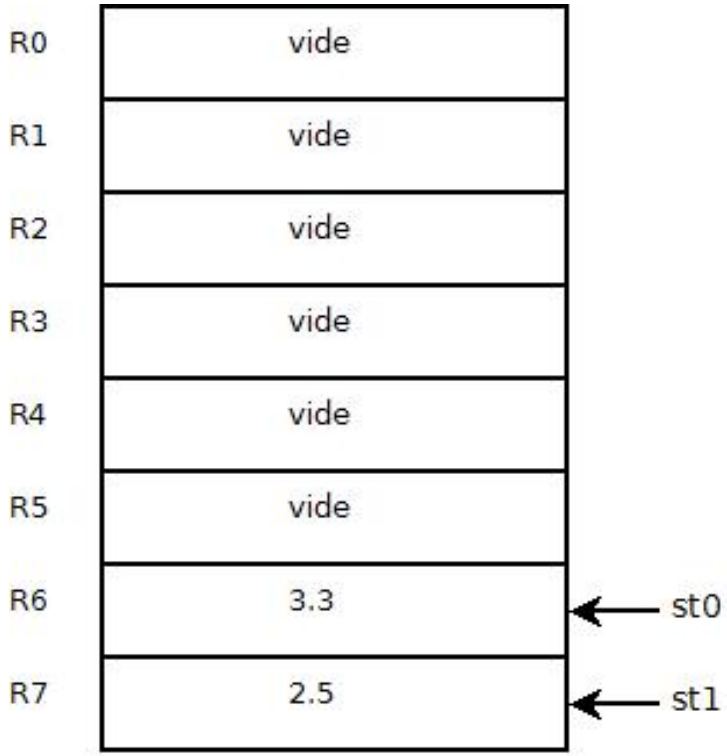
La pile du 8087 : exemple 2

- Partons de la pile obtenue dans l'exemple précédent
- POP



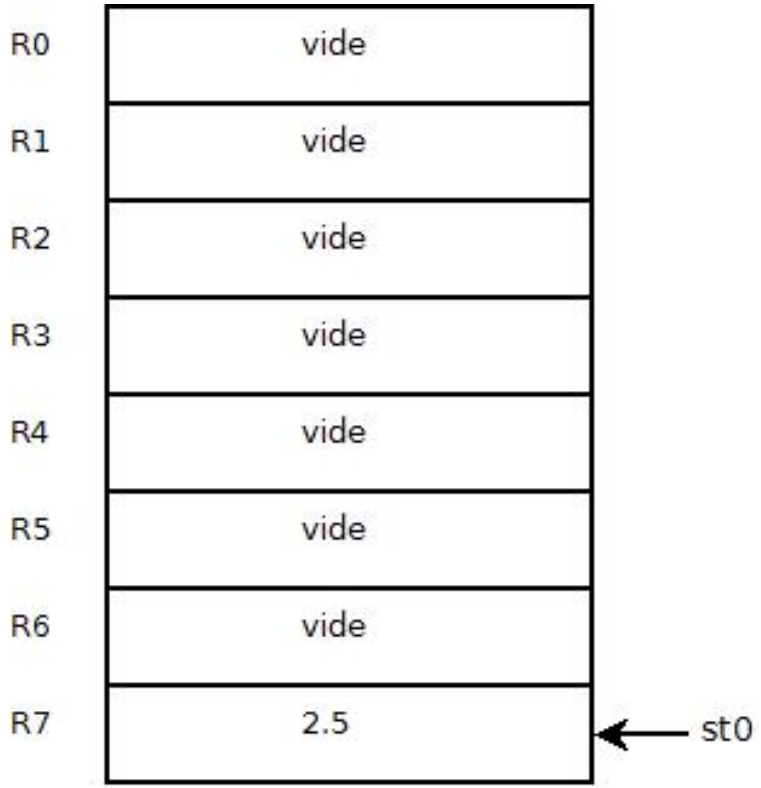
La pile du 8087 : exemple 2

- Partons de la pile obtenue dans l'exemple précédent
- POP
- POP



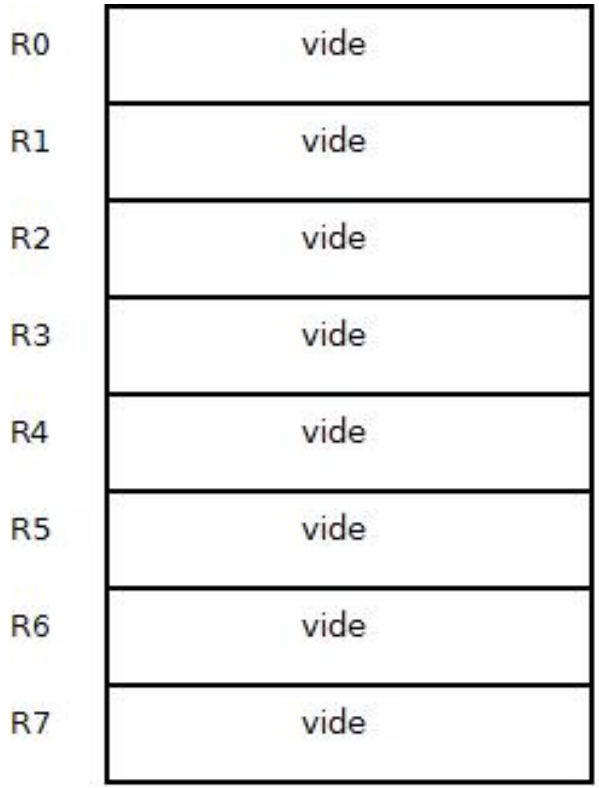
La pile du 8087 : exemple 2

- Partons de la pile obtenue dans l'exemple précédent
- POP
- POP
- POP



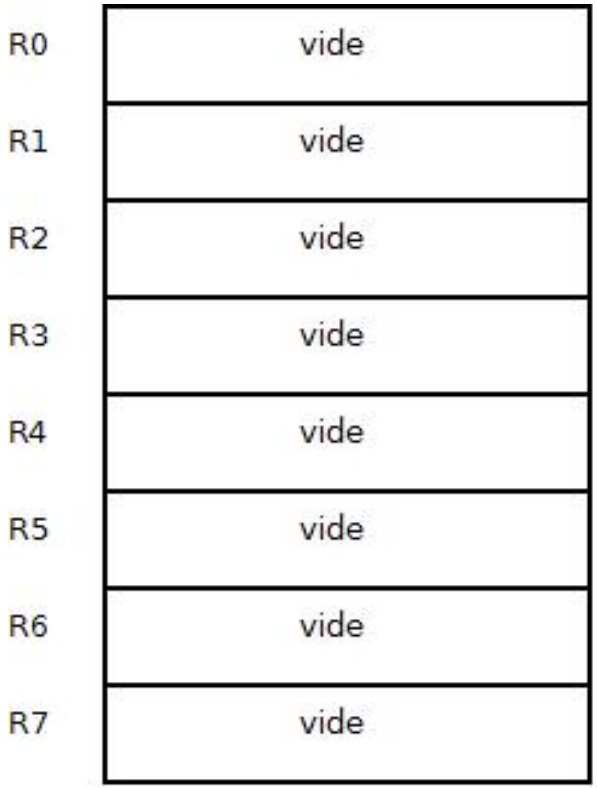
La pile du 8087 : exemple 2

- Partons de la pile obtenue dans l'exemple précédent
- POP
- POP
- POP
- POP



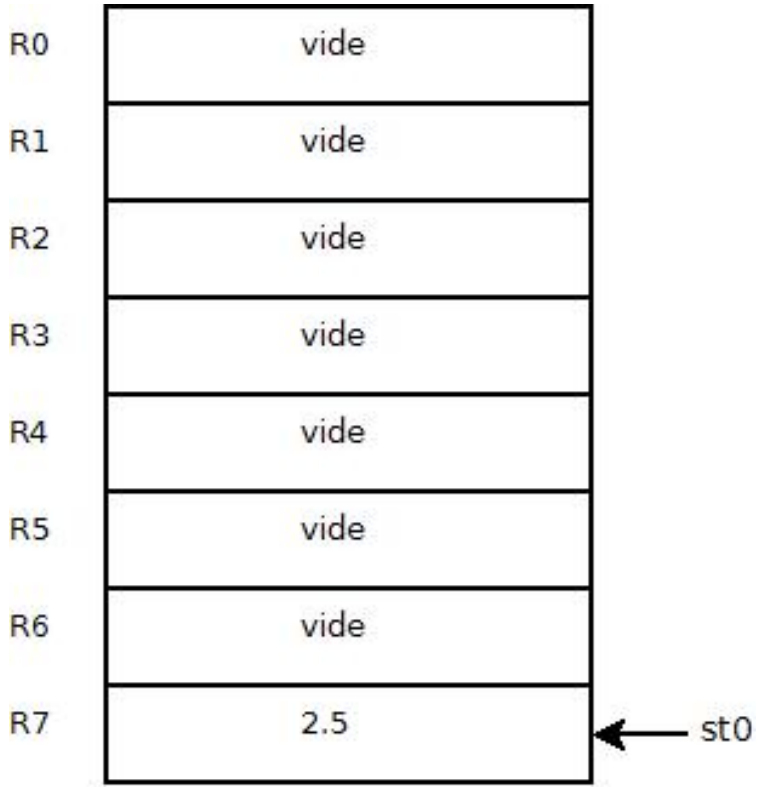
La pile du 8087 : exemple 3

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



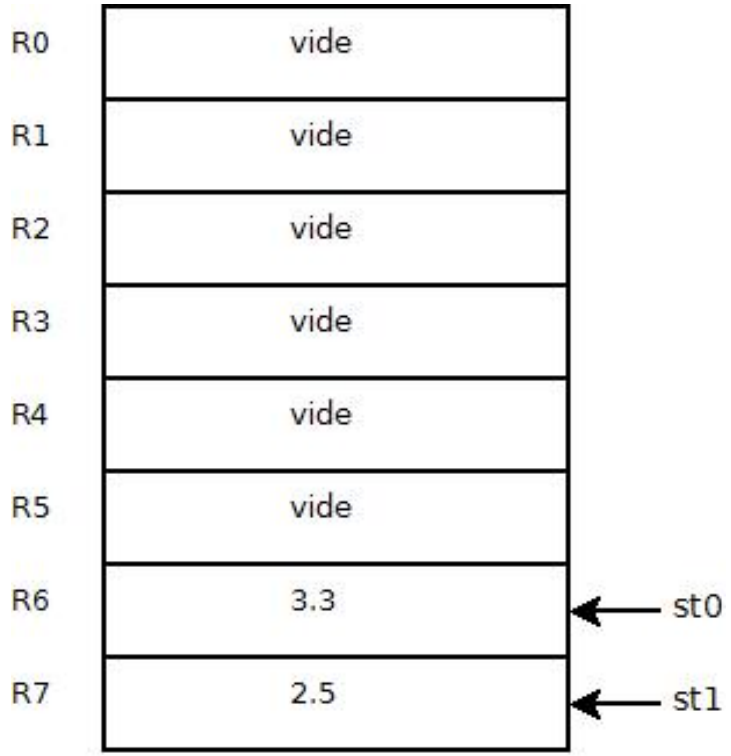
La pile du 8087 : exemple 3

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



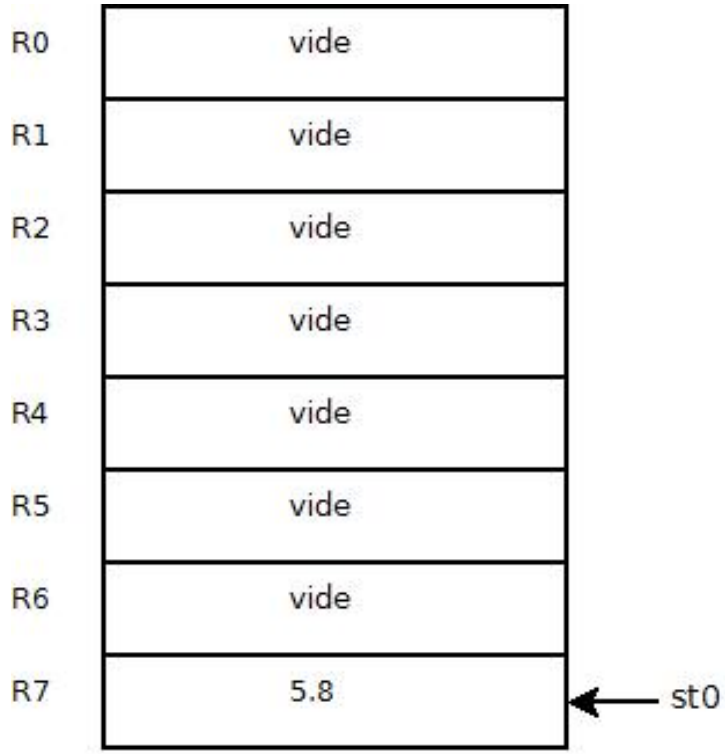
La pile du 8087 : exemple 3

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



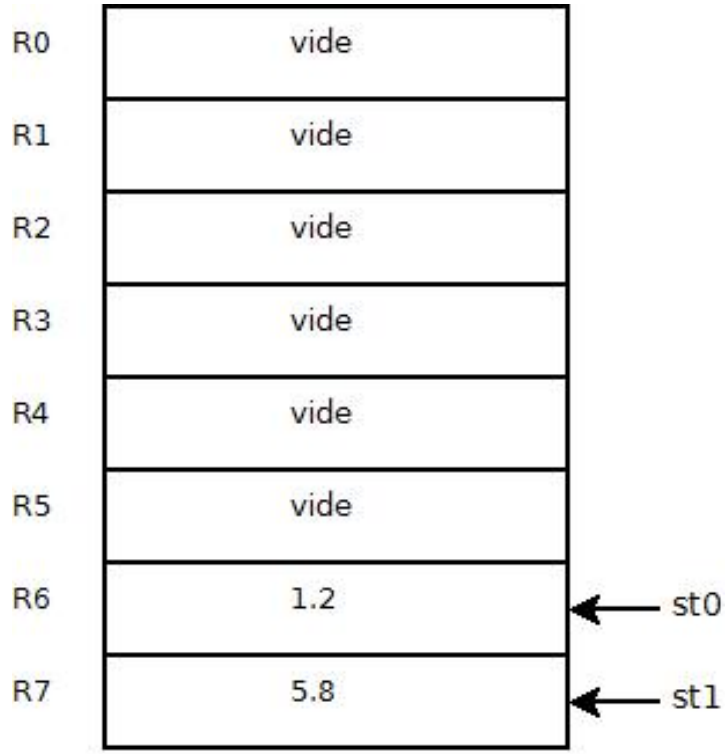
La pile du 8087 : exemple 3

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



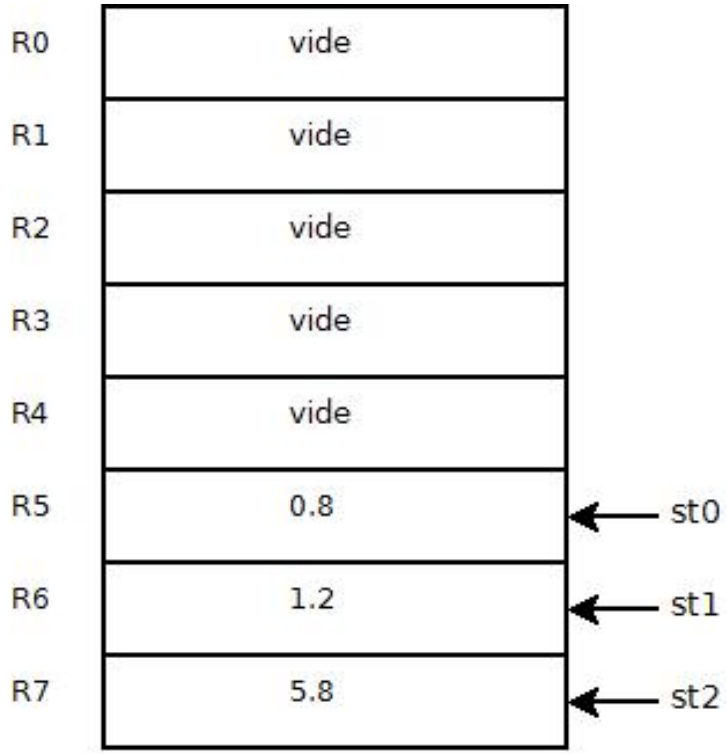
La pile du 8087 : exemple 3

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



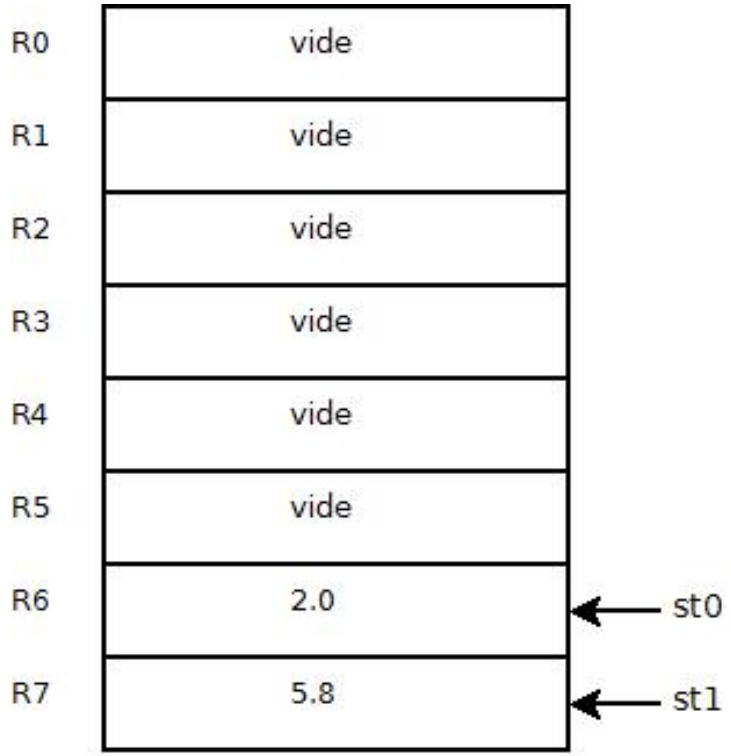
La pile du 8087 : exemple 3

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



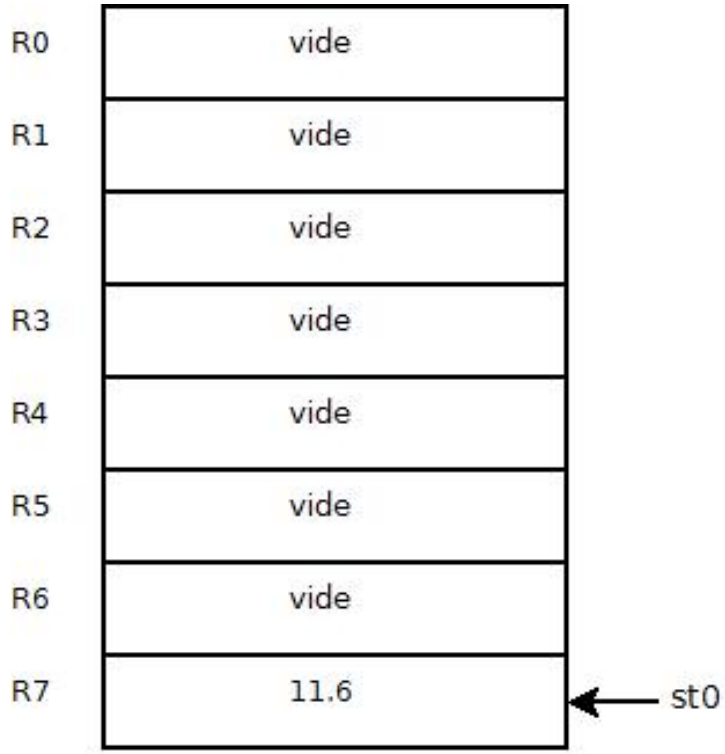
La pile du 8087 : exemple 3

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



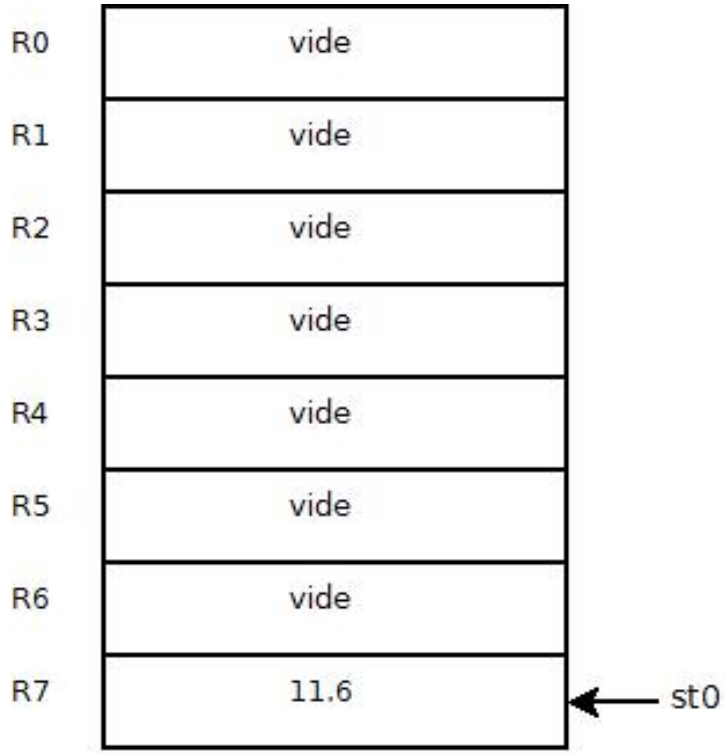
La pile du 8087 : exemple 3

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



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*

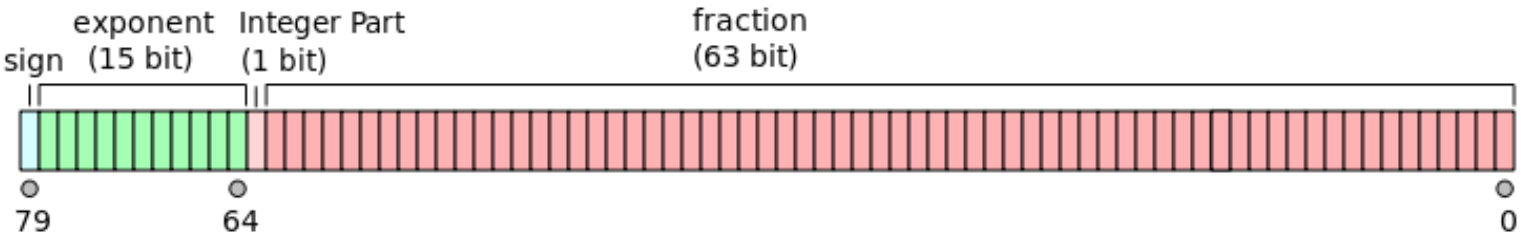


L'encodage des réels sous Intel 8087

- Les registres *st0* à *st7* contiennent des réels de 80 bits
- Valeurs possibles dans les registres :
  - Nombre réel (80 bits)
  - « `vide` »
  - « `infini` »
  - « `NAN` »
- `tword` = 10 octets (t = *ten*)
  - Même usage que `byte`, `word`, `dword`

L'encodage des réels sous Intel 8087

- 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)

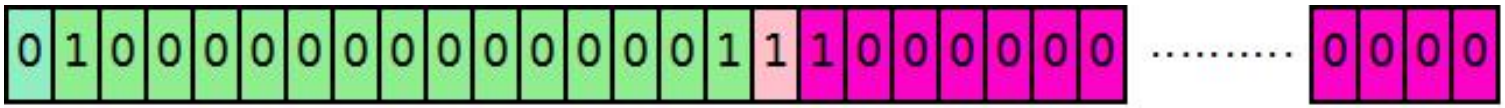


- Formule :

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

L'encodage des réels sous Intel 8087

- Exemple : $4001C000000000000000$

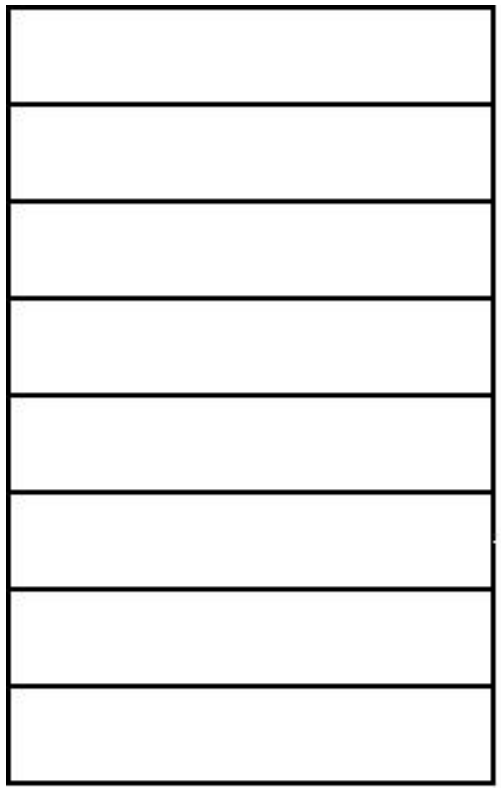


- $s = 0$
- $e = 1000000000000001_2 = 16385_{10}$
- $m = 1.100000....0_2 = 1.5_{10}$
- Formule :

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

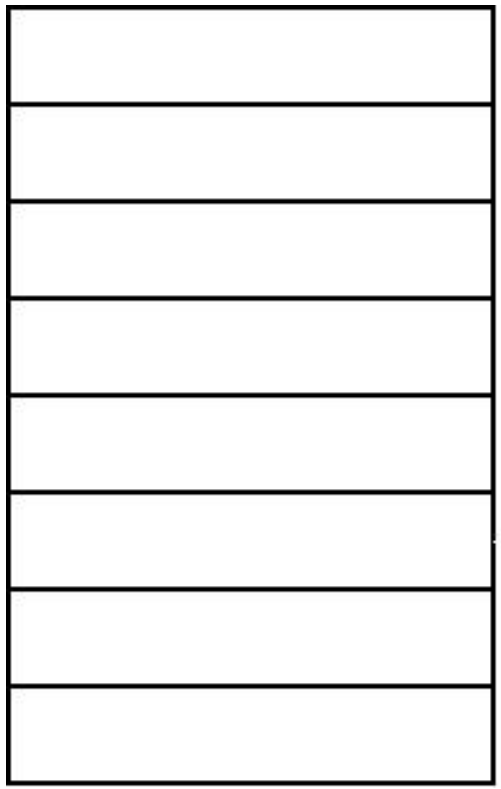
Instructions du 8087 : le PUSH

- `FLD` = Floating-point LoaD
- Syntaxe :
  - `FLD param`
  - `param` est pushé au sommet
  - `param` = immédiat, mémoire, st*i*



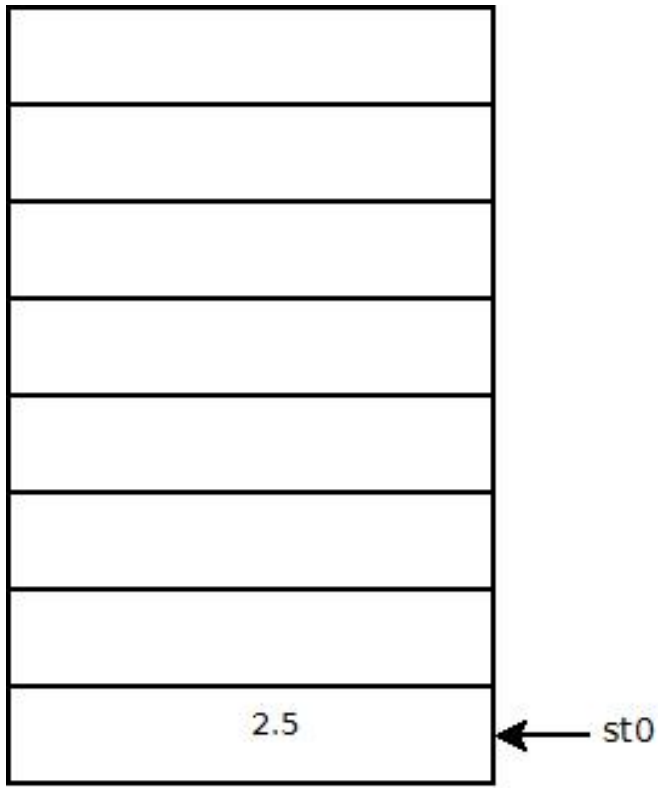
Instructions du 8087 : le PUSH

- `FLD` = Floating-point LoaD
- Syntaxe :
  - `FLD param`
  - `param` est pushé au sommet
  - `param` = immédiat, mémoire, st*i*
- Exemple :



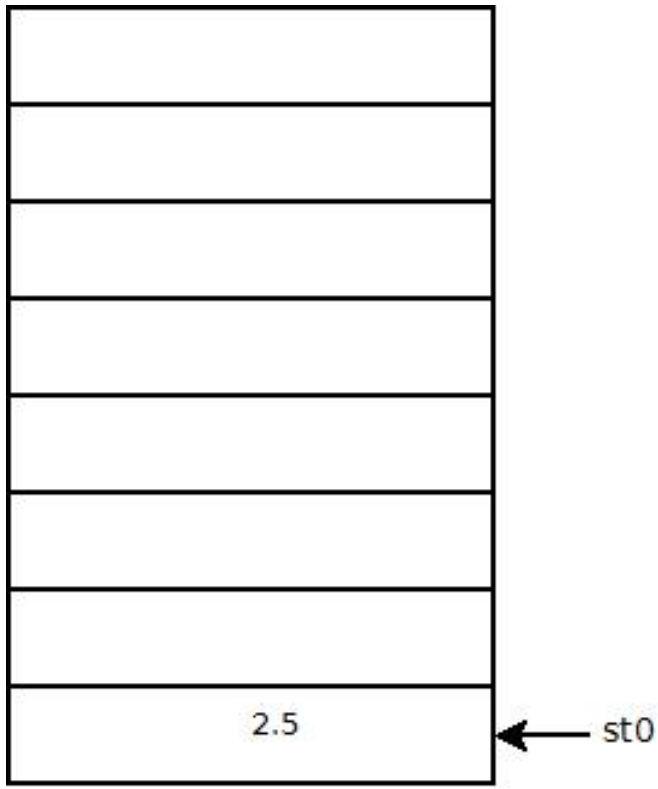
Instructions du 8087 : le PUSH

- `FLD` = Floating-point LoaD
- Syntaxe :
  - `FLD param`
  - `param` est pushé au sommet
  - `param` = immédiat, mémoire, st*i*
- Exemple :
  - `FLD tword 2.5`



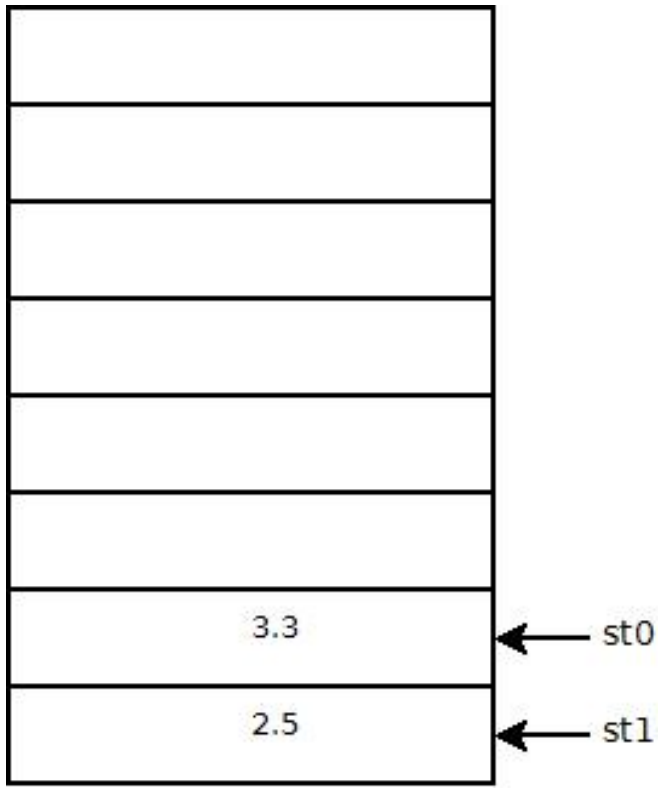
Instructions du 8087 : le PUSH

- `FLD` = Floating-point LoaD
- Syntaxe :
  - `FLD param`
  - `param` est pushé au sommet
  - `param` = immédiat, mémoire, st*i*
- Exemple :
  - `FLD tword 2.5`
    - `st0` ← 2.5



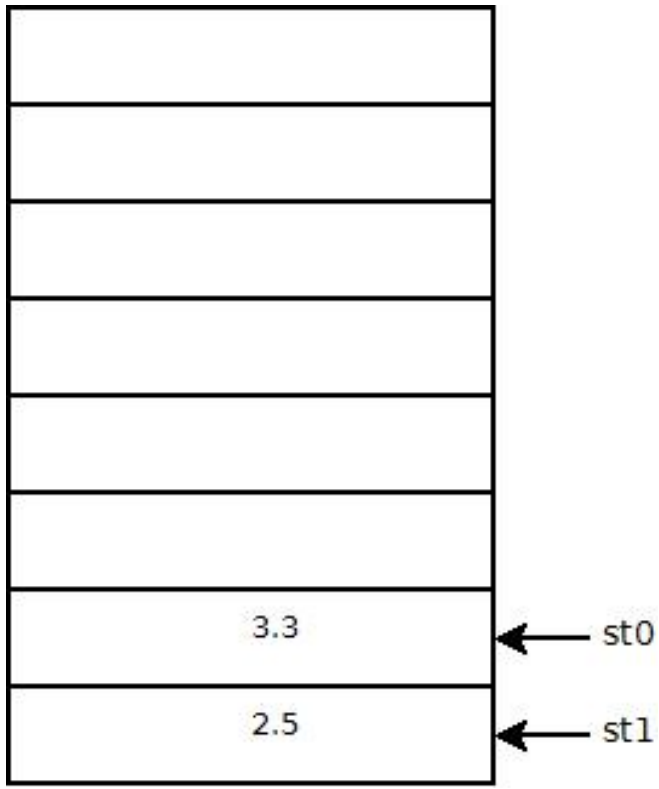
Instructions du 8087 : le PUSH

- `FLD` = Floating-point LoaD
- Syntaxe :
  - `FLD param`
  - `param` est pushé au sommet
  - `param` = immédiat, mémoire, st*i*
- Exemple :
  - `FLD tword 2.5`
    - `st0` ← 2.5
  - `FLD tword 3.3`



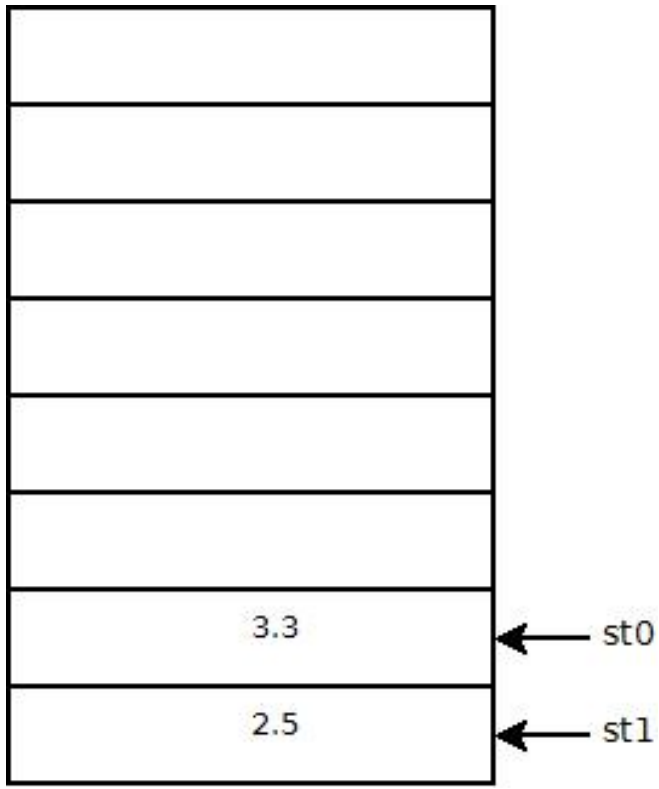
Instructions du 8087 : le PUSH

- `FLD` = Floating-point LoaD
- Syntaxe :
  - `FLD param`
  - `param` est pushé au sommet
  - `param` = immédiat, mémoire, st*i*
- Exemple :
  - `FLD tword 2.5`
    - `st0` ← 2.5
  - `FLD tword 3.3`
    - `st1` ← 2.5



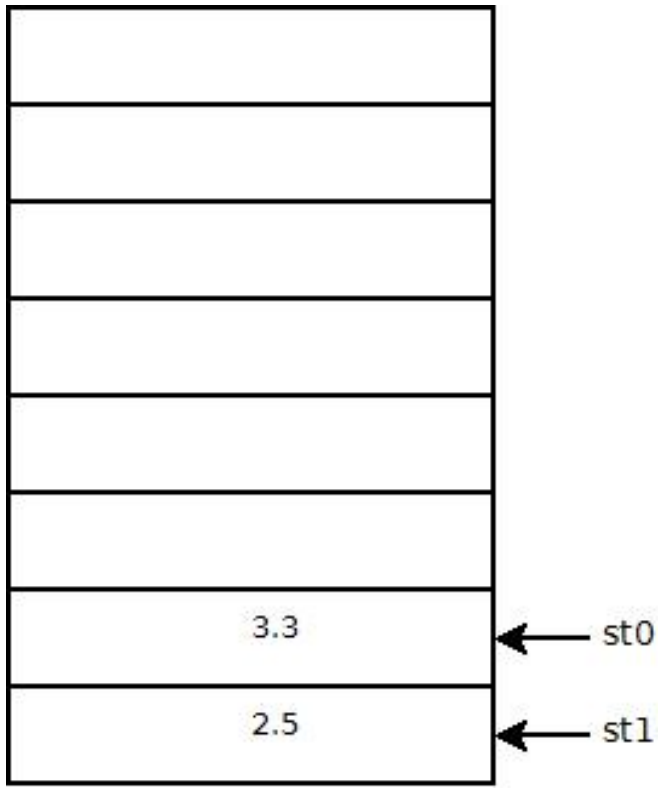
Instructions du 8087 : le PUSH

- `FLD` = Floating-point LoaD
- Syntaxe :
  - `FLD param`
  - `param` est pushé au sommet
  - `param` = immédiat, mémoire, st*i*
- Exemple :
  - `FLD tword 2.5`
    - `st0` ← 2.5
  - `FLD tword 3.3`
    - `st1` ← 2.5
    - `st0` ← 3.3



Instructions du 8087 : le PUSH

- `FLD` = Floating-point LoaD
- Syntaxe :
  - `FLD param`
  - `param` est pushé au sommet
  - `param` = immédiat, mémoire, st*i*
- Exemple :
  - `FLD tword 2.5`
    - `st0` ← 2.5
  - `FLD tword 3.3`
    - `st1` ← 2.5
    - `st0` ← 3.3

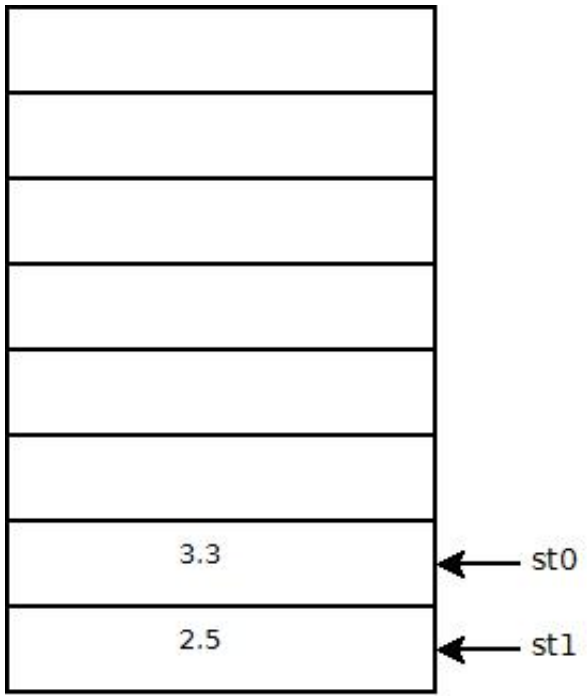


Instructions du 8087 : le PUSH

- Variantes de `FLD` :
  - `FLDZ` : PUSH 0.0
  - `FLD1` : PUSH 1.0
  - `FLDPI` : PUSH π

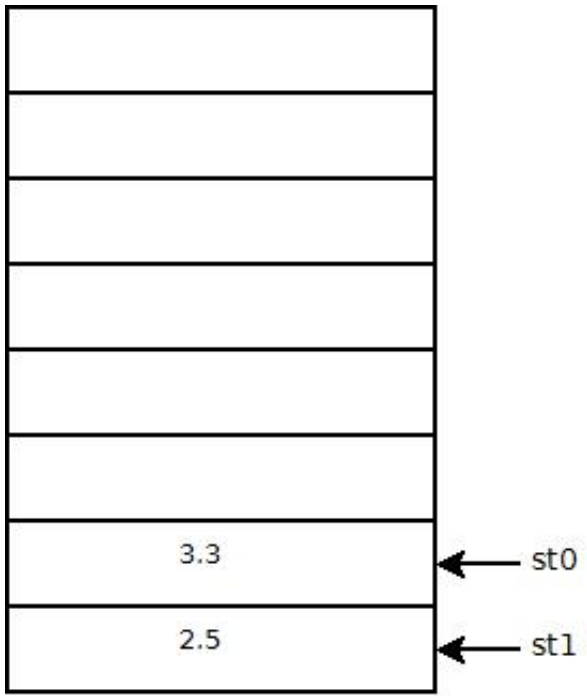
Instructions du 8087 : le TOP

- `FST` = Floating-point STore
- Syntaxe : `FST param`
- Effet :
  - `param` $\leftarrow$ sommet de pile



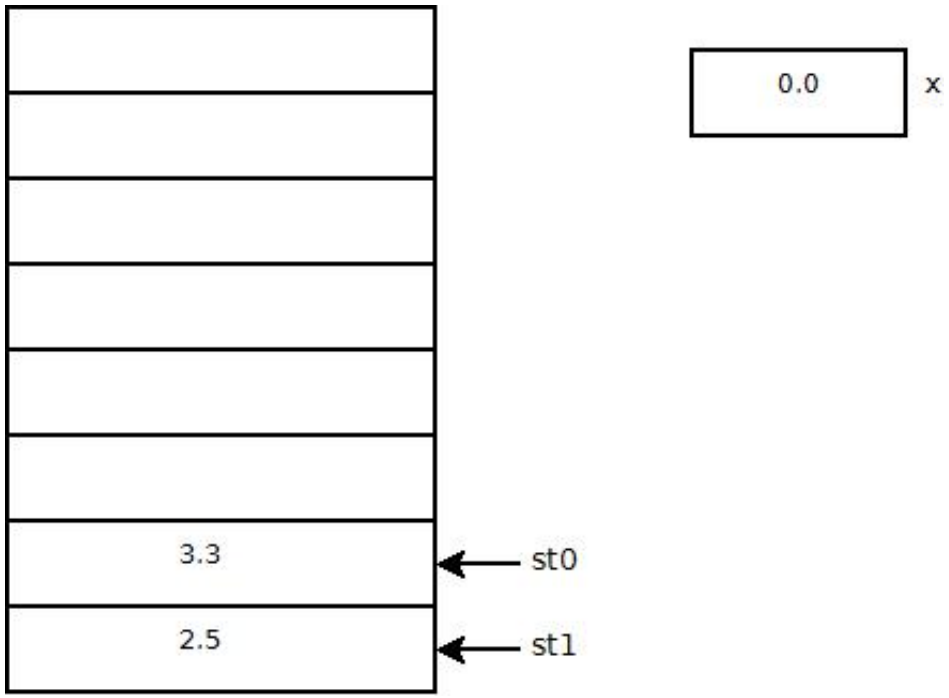
Instructions du 8087 : le TOP

- `FST` = Floating-point STore
- Syntaxe : `FST param`
- Effet :
  - `param` $\leftarrow$ sommet de pile
- Exemple :



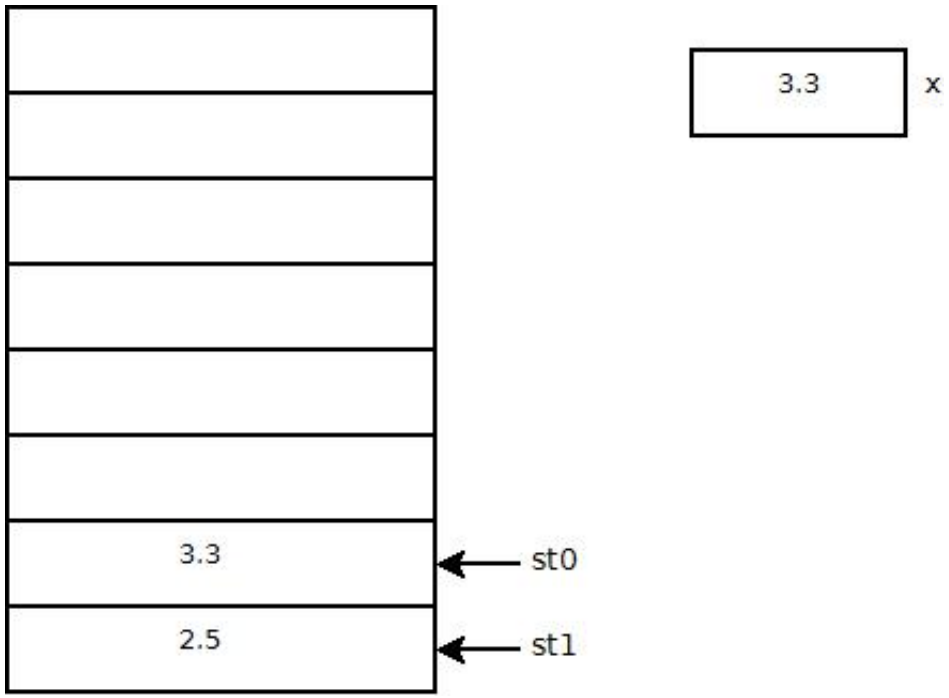
Instructions du 8087 : le TOP

- `FST` = Floating-point STore
- Syntaxe : `FST param`
- Effet :
  - `param` ← sommet de pile
- Exemple :
  - `x DT 0.0`



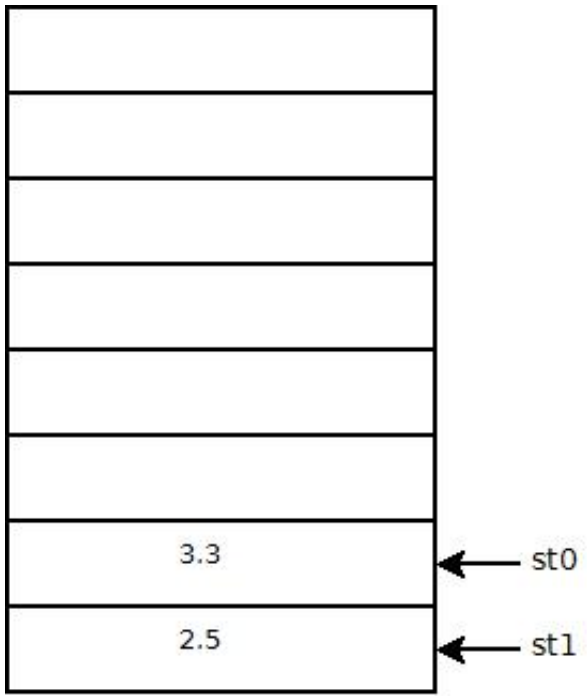
Instructions du 8087 : le TOP

- `FST` = Floating-point STore
- Syntaxe : `FST param`
- Effet :
  - `param` ← sommet de pile
- Exemple :
  - `x DT 0.0`
  - `FST tword [x]`



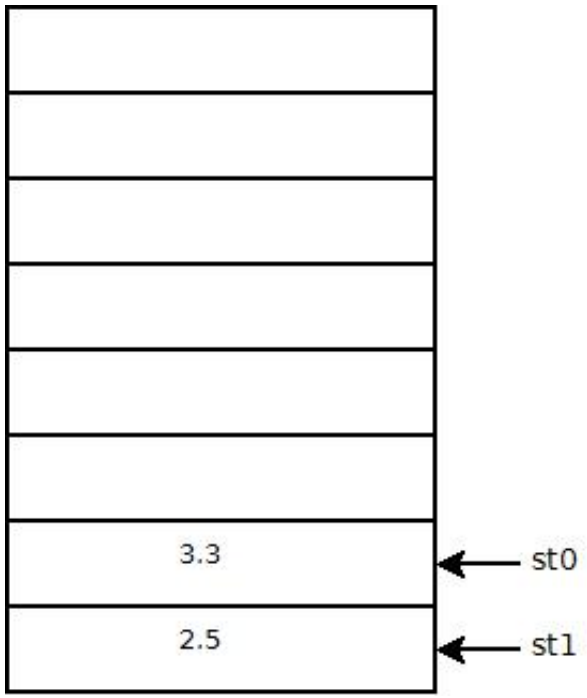
Instructions du 8087 : le POP

- `FSTP` = Floating-point STore Pop
- Syntaxe : `FSTP param`
- Effet :
  - `param` ← sommet de pile
  - POP



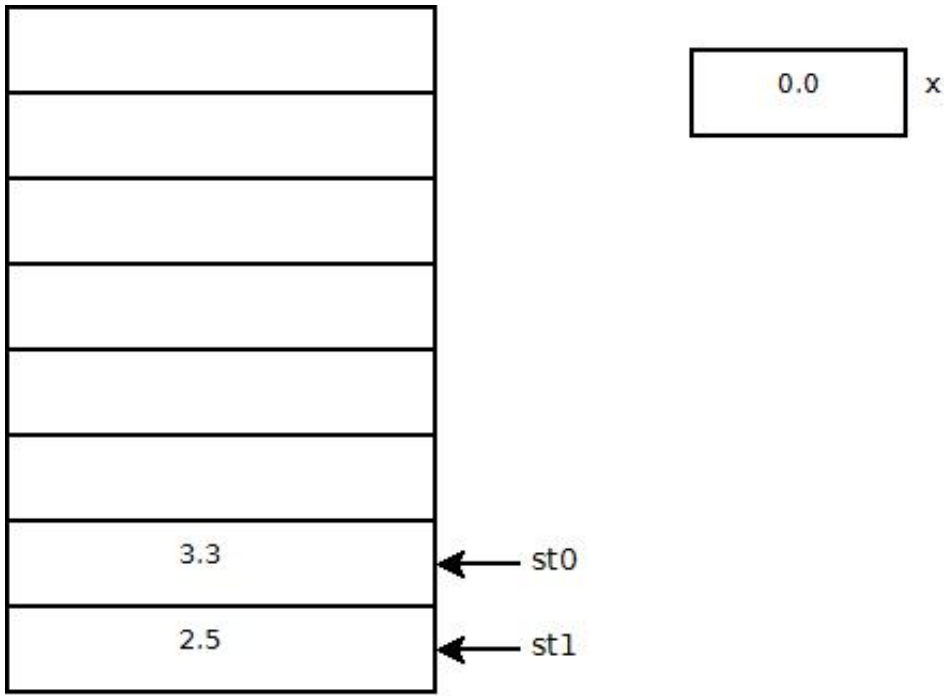
Instructions du 8087 : le POP

- `FSTP` = Floating-point STore Pop
- Syntaxe : `FSTP param`
- Effet :
  - `param` ← sommet de pile
  - POP
- Exemple :



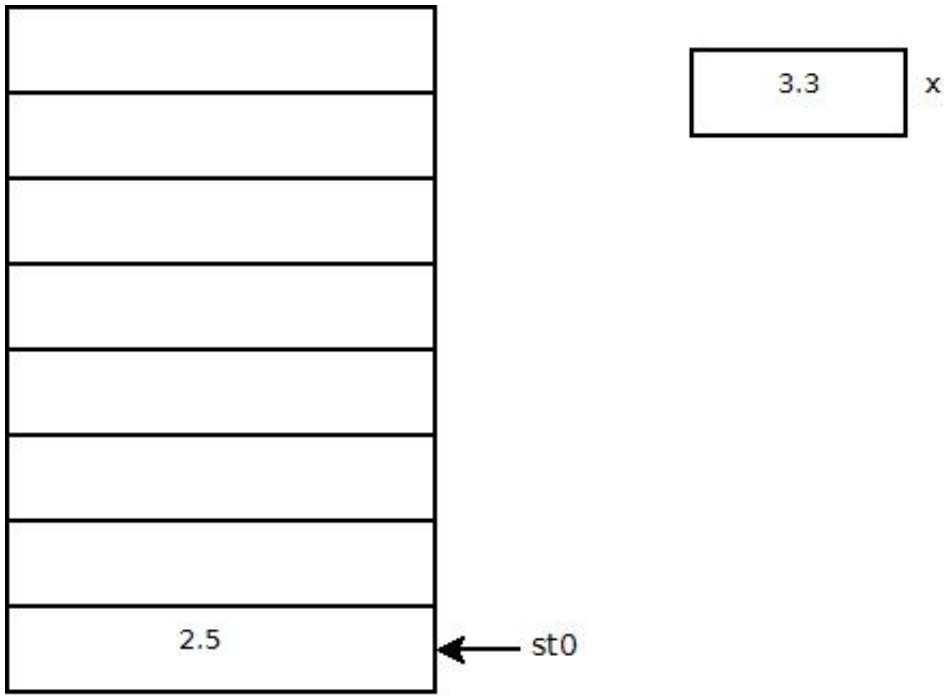
Instructions du 8087 : le POP

- `FSTP` = Floating-point STore Pop
- Syntaxe : `FSTP param`
- Effet :
  - `param` ← sommet de pile
  - POP
- Exemple :
  - `x DT 0.0`



Instructions du 8087 : le POP

- `FSTP` = Floating-point STore Pop
- Syntaxe : `FSTP param`
- Effet :
  - `param` ← sommet de pile
  - POP
- Exemple :
  - `x DT 0.0`
  - `FSTP tword [x]`

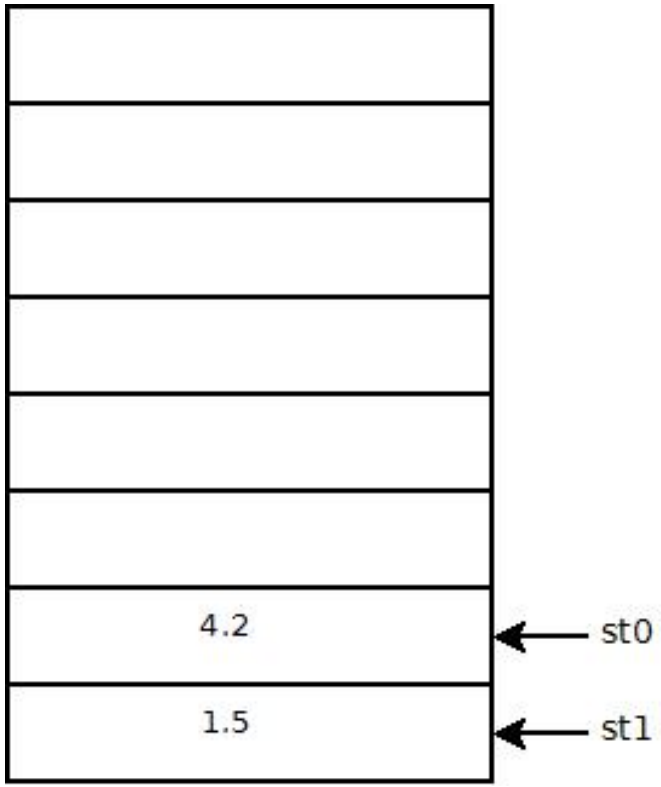


Opérations arithmétiques du 8087

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

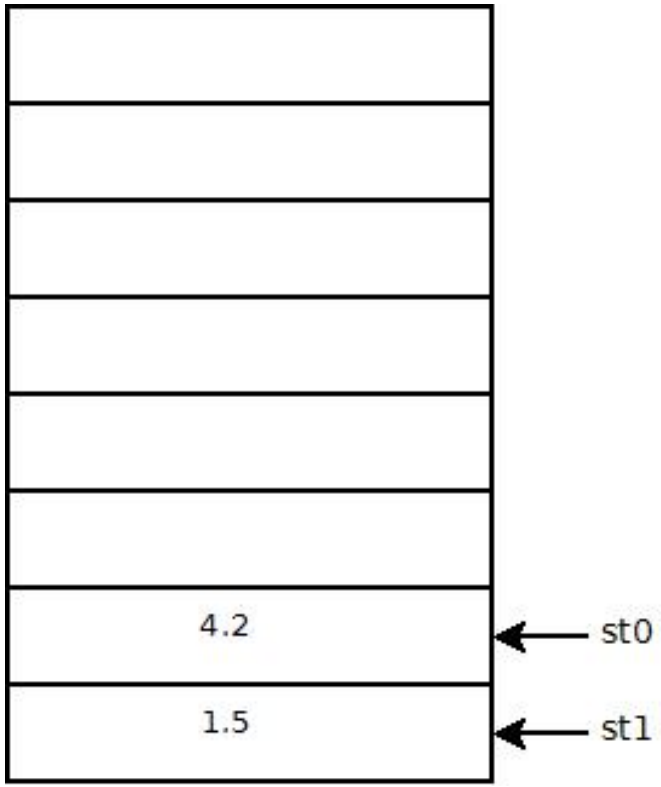
Addition

- `FADD` = Floating-point ADD
- `FADD st`i`, st`j
  - `st`$i \leftarrow$ `st`i + `st`j



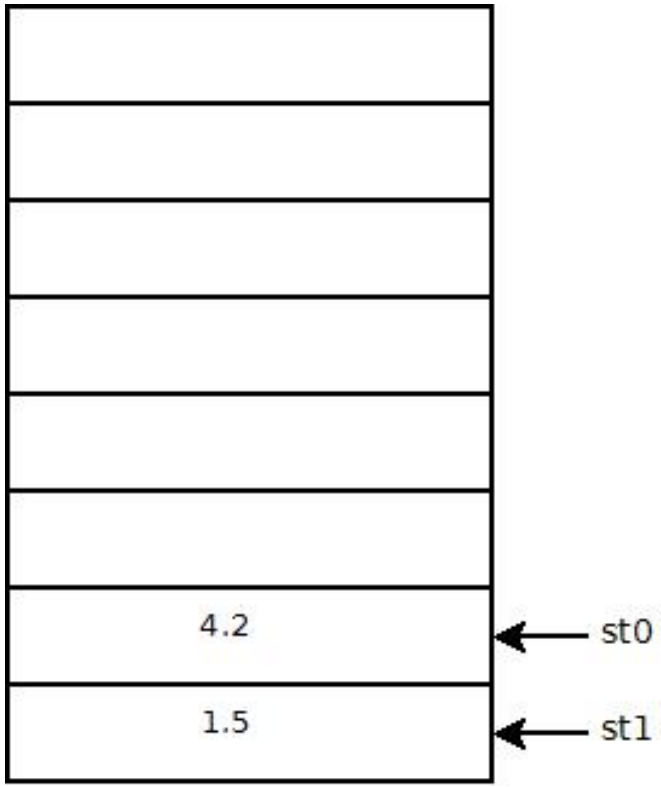
Addition

- `FADD` = Floating-point ADD
- `FADD st`i`, st`j
  - `st`i $\leftarrow$ `st`i + `st`j
- Exemple :



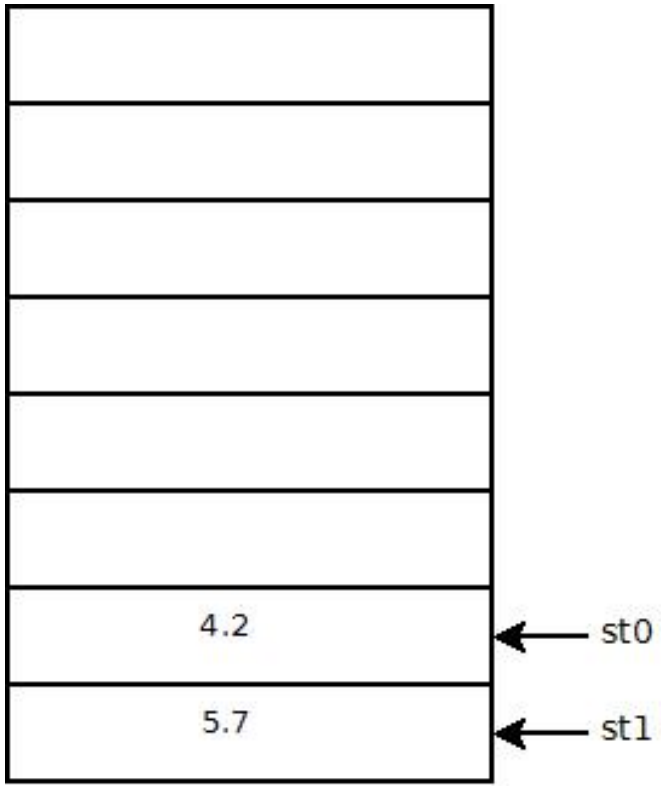
Addition

- `FADD` = Floating-point ADD
- `FADD st`i`, st`j
  - `st`i $\leftarrow$ `st`i + `st`j
- Exemple :
  - `FADD st1, st0`



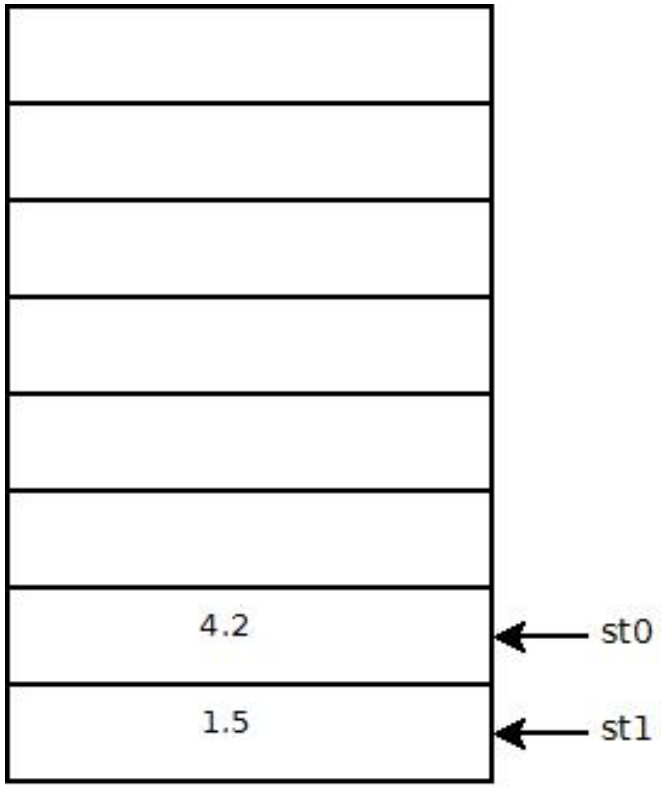
Addition

- `FADD` = Floating-point ADD
- `FADD st`i`, st`j
  - `st`i $\leftarrow$ `st`i + `st`j
- Exemple :
  - `FADD st1, st0`
  - `st1` $\leftarrow$ `st1` + `st0`



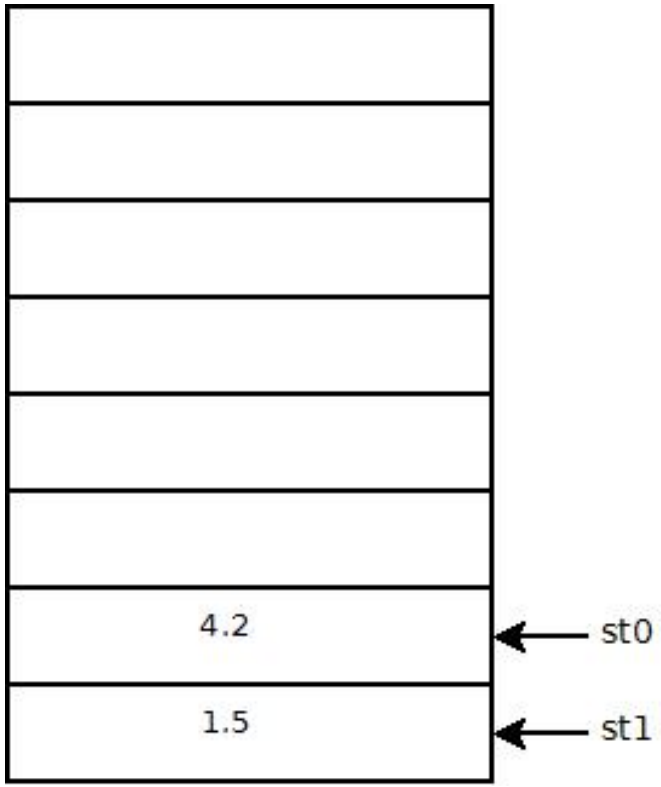
Addition avec pop

- `FADDP` = Floating-point ADD Pop
- `FADDP st`i`, st`j
  - `st`i $\leftarrow$ `st`i + `st`j
  - POP



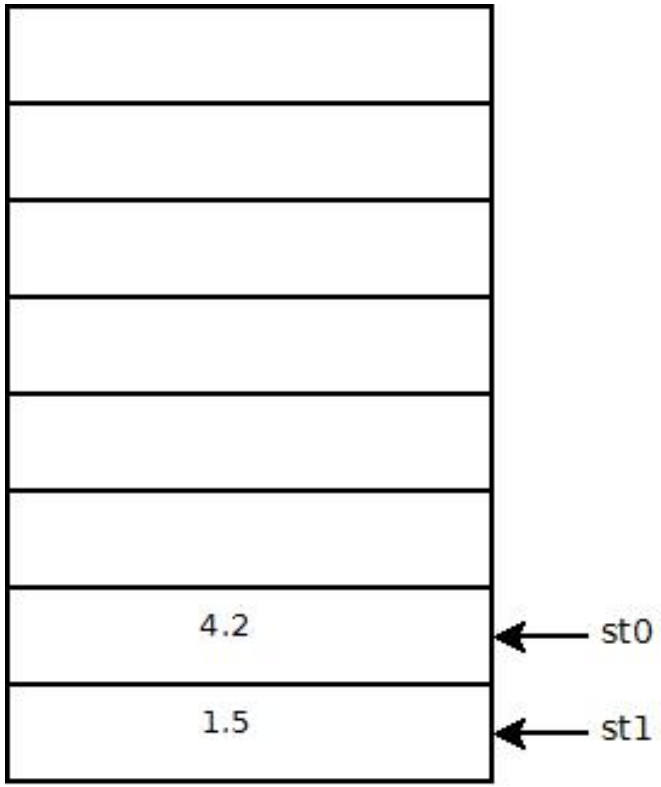
Addition avec pop

- `FADDP` = Floating-point ADD Pop
- `FADDP st`i`, st`j
  - `st`i $\leftarrow$ `st`i + `st`j
  - POP
- Exemple :



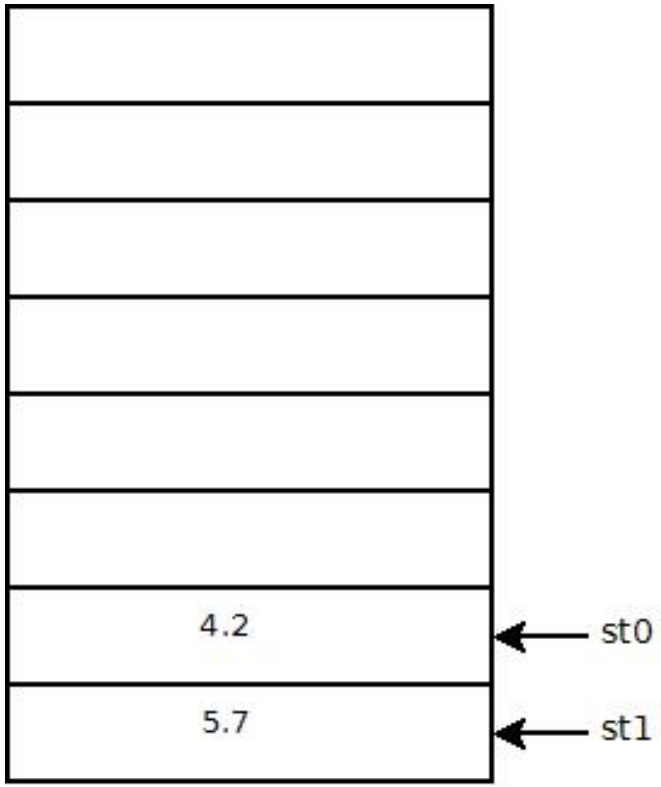
Addition avec pop

- `FADDP` = Floating-point ADD Pop
- `FADDP st`i`, st`j
  - `st`$i \leftarrow$ `st`i + `st`j
  - POP
- Exemple :
  - `FADDP st1, st0`



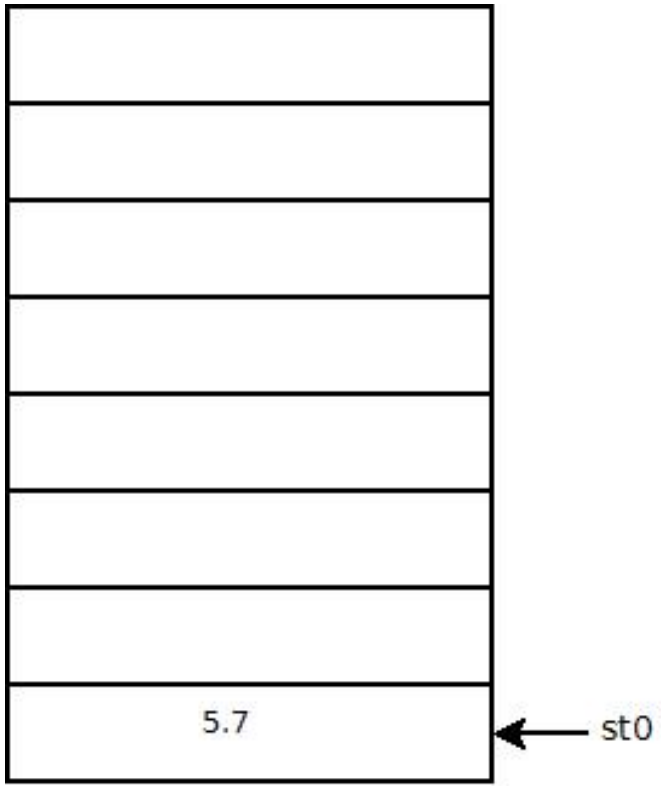
Addition avec pop

- `FADDP` = Floating-point ADD Pop
- `FADDP sti, stj`
  - `sti` $\leftarrow$ `sti` + `stj`
  - POP
- Exemple :
  - `FADDP st1, st0`
  - `st1` $\leftarrow$ `st1` + `st0`



Addition avec pop

- `FADDP` = Floating-point ADD Pop
- `FADDP sti, stj`
  - $\texttt{st}i \leftarrow \texttt{st}i + \texttt{st}j$
  - POP
- Exemple :
  - `FADDP st1, st0`
  - $\texttt{st1} \leftarrow \texttt{st1} + \texttt{st0}$
  - POP



Addition : raccourcis

- `FADDP st`i
  - raccourci pour `FADDP st`i`, st0`
- `FADDP`
  - raccourci pour `FADDP st1, st0`

Addition : exemple 1

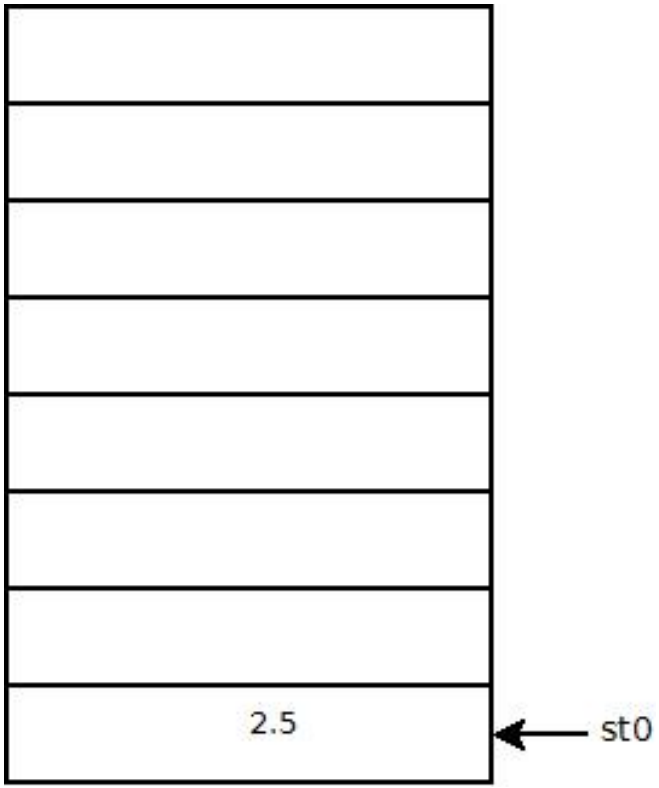
- $2.5 + 3.3$

Addition : exemple 1

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

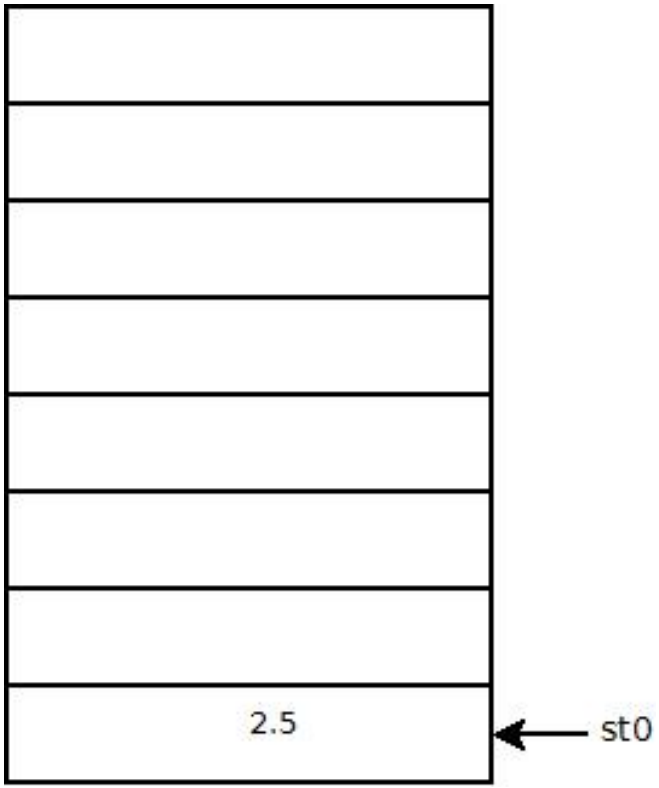
Addition : exemple 1

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



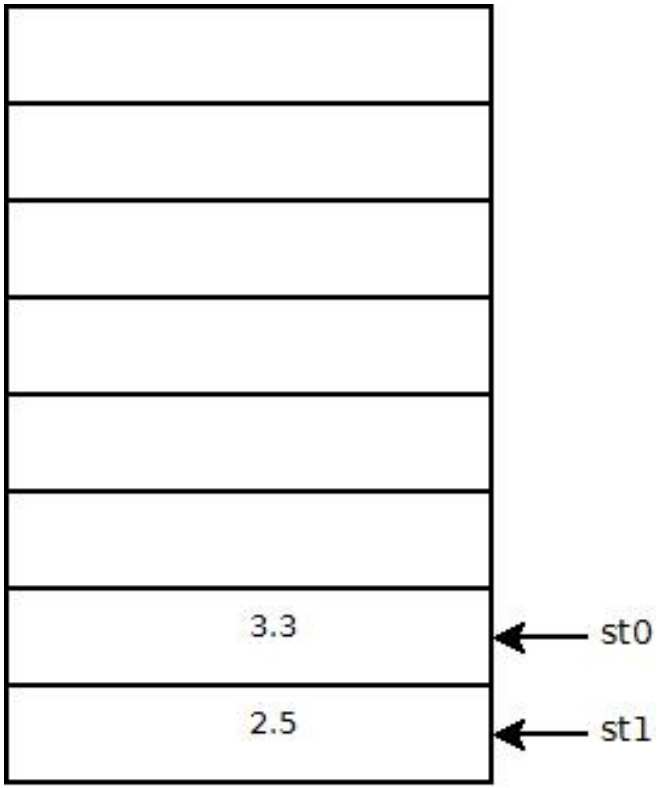
Addition : exemple 1

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



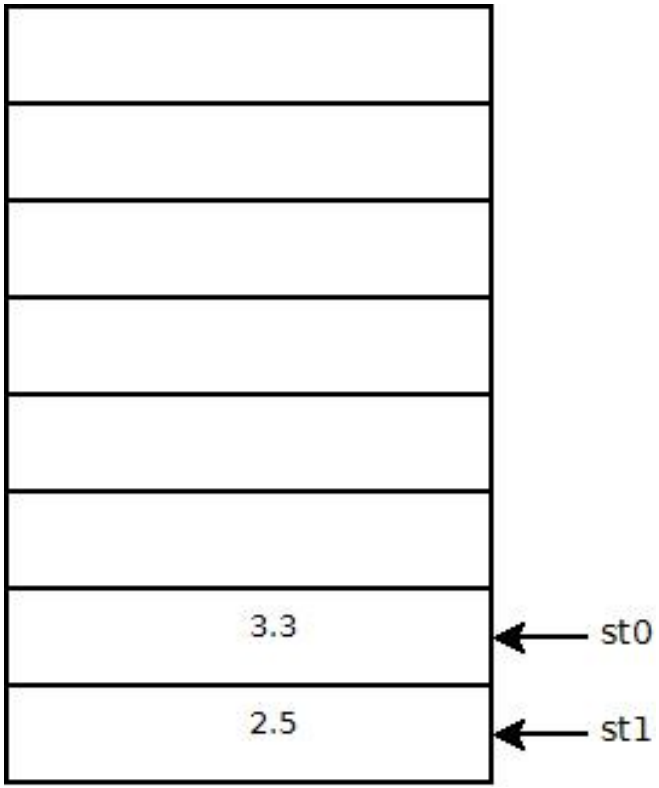
Addition : exemple 1

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



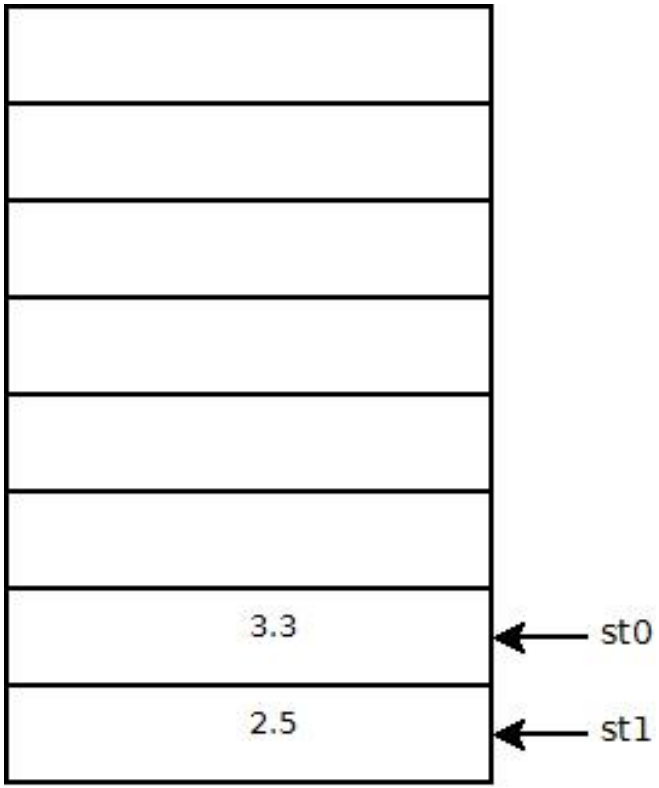
Addition : exemple 1

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



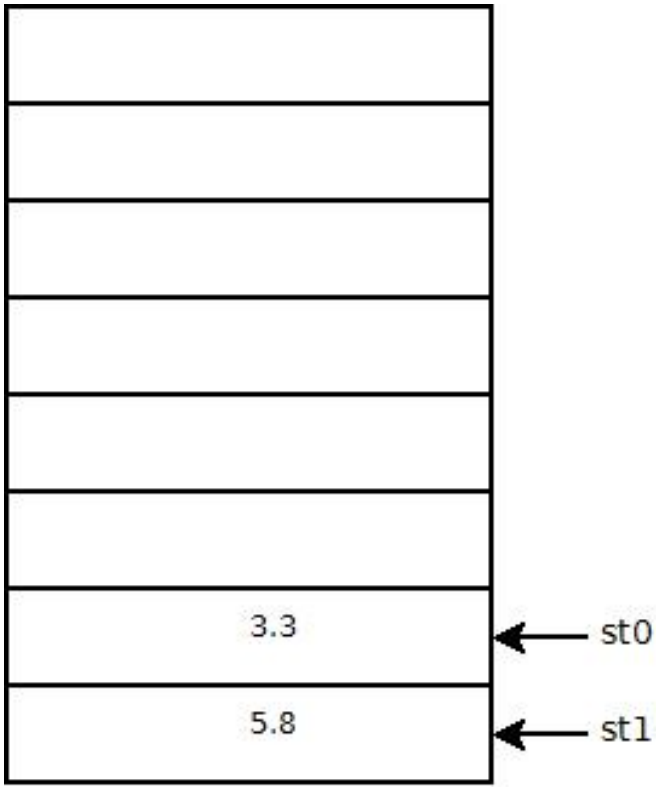
Addition : exemple 1

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



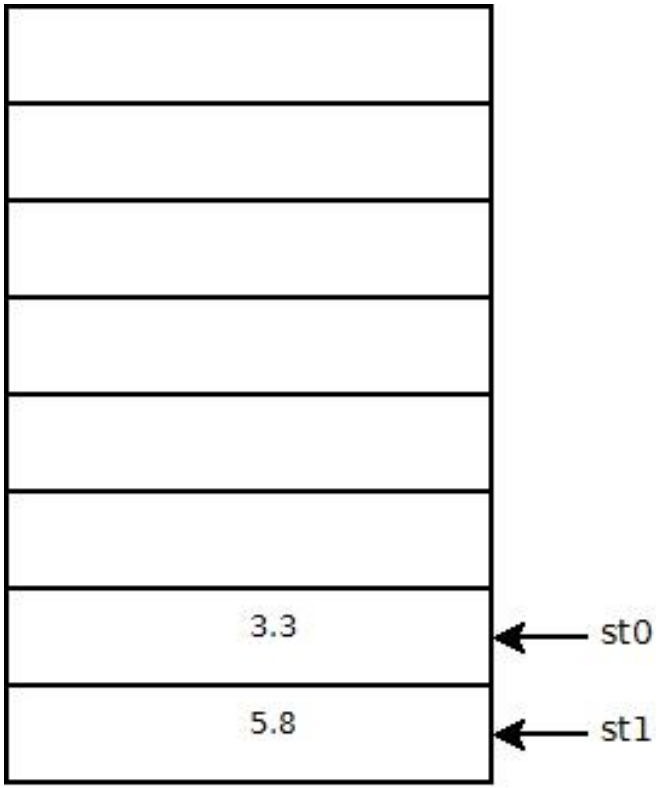
Addition : exemple 1

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



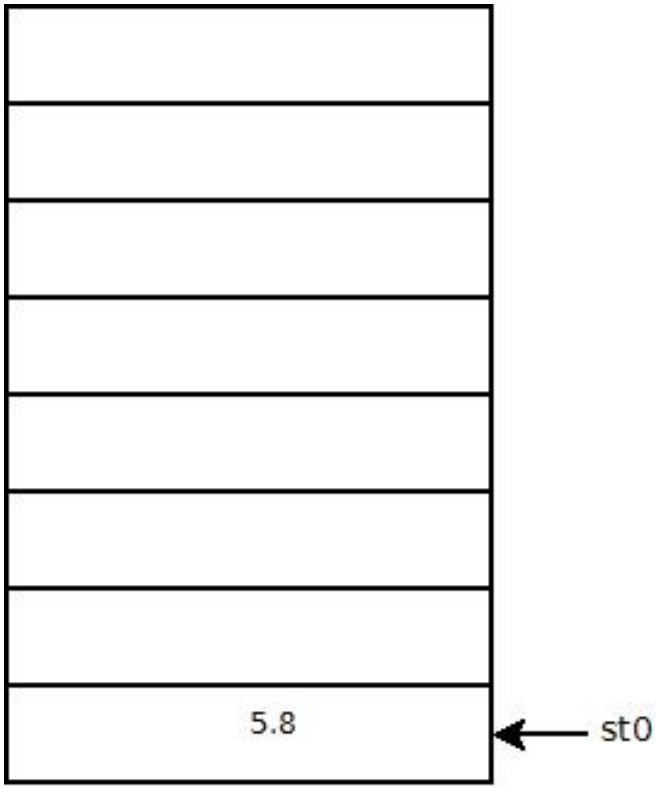
Addition : exemple 1

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



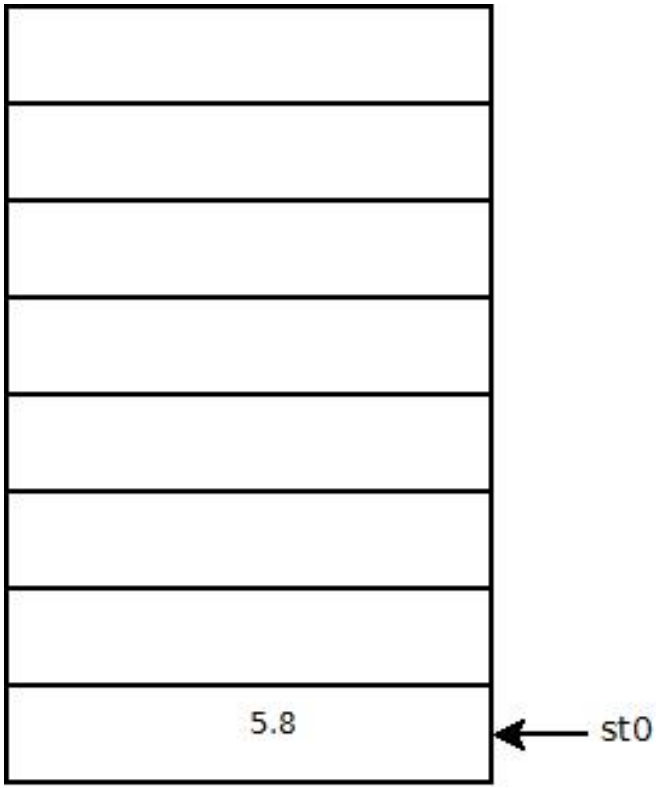
Addition : exemple 1

- $2.5 + 3.3$
- `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

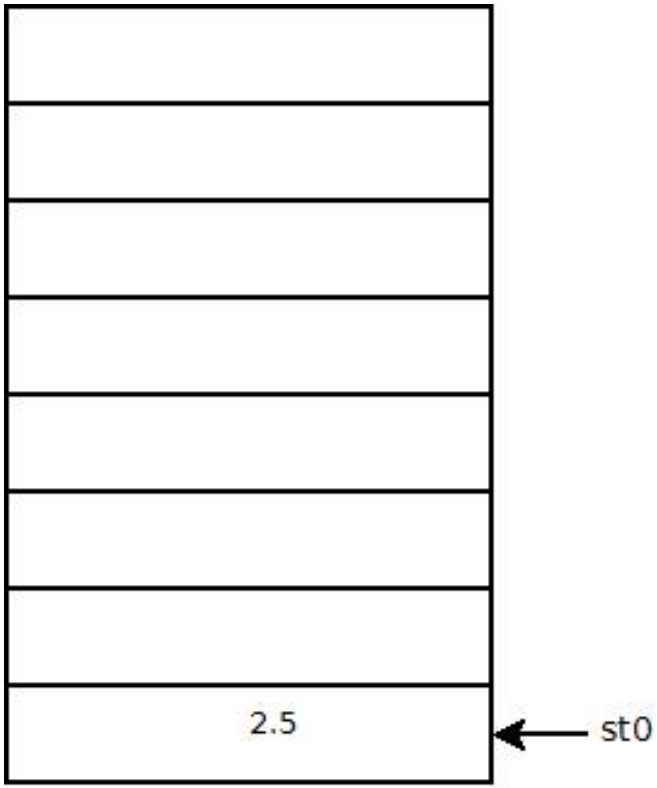
- $2.5 + 3.3 + 4.1$

Addition : exemple 2

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

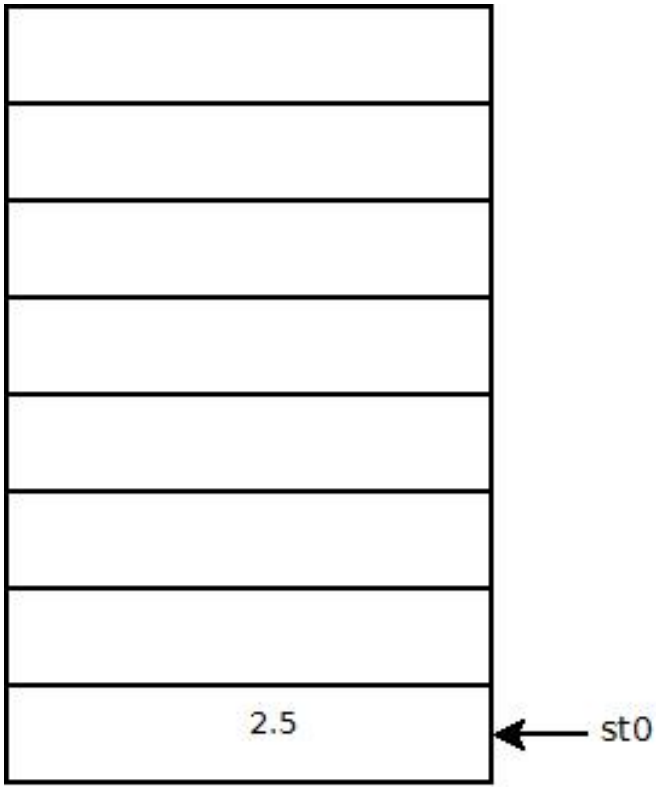
Addition : exemple 2

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



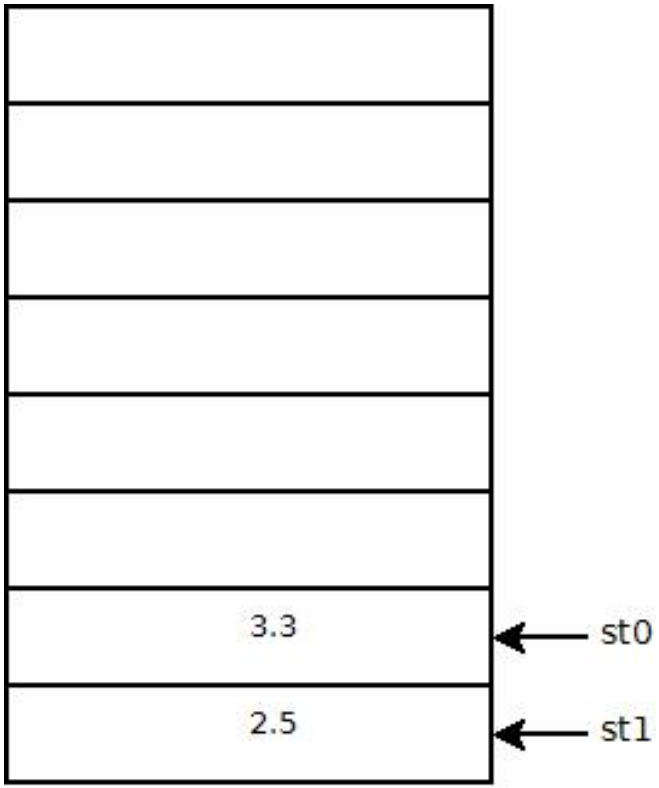
Addition : exemple 2

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



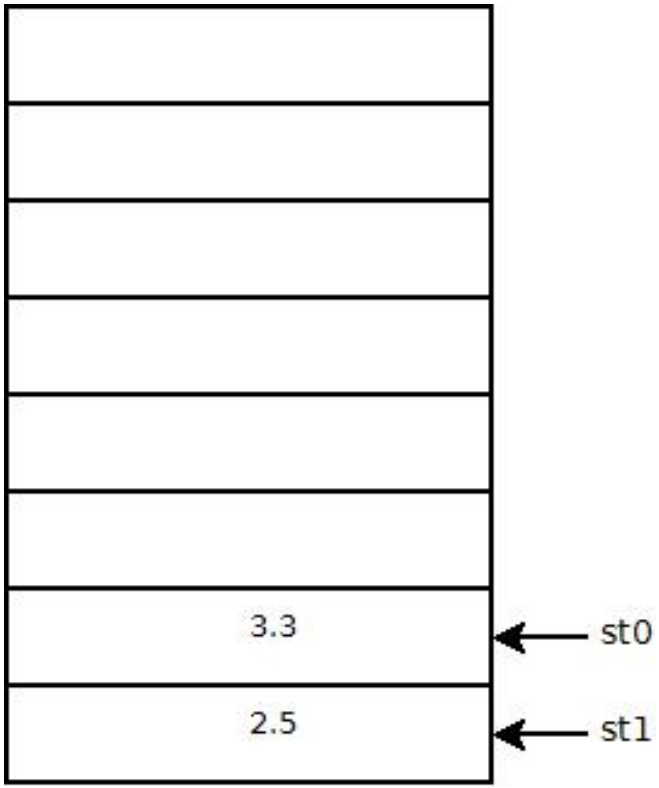
Addition : exemple 2

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



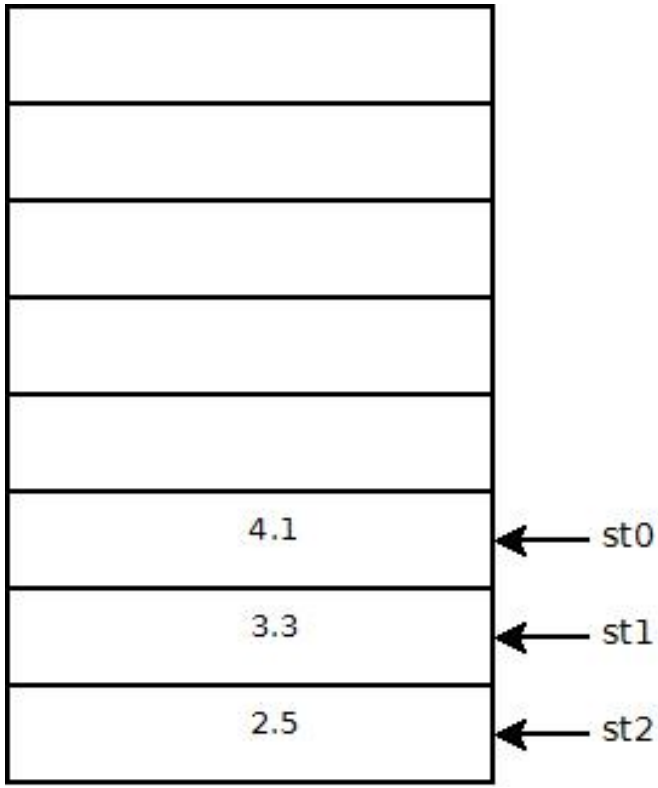
Addition : exemple 2

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



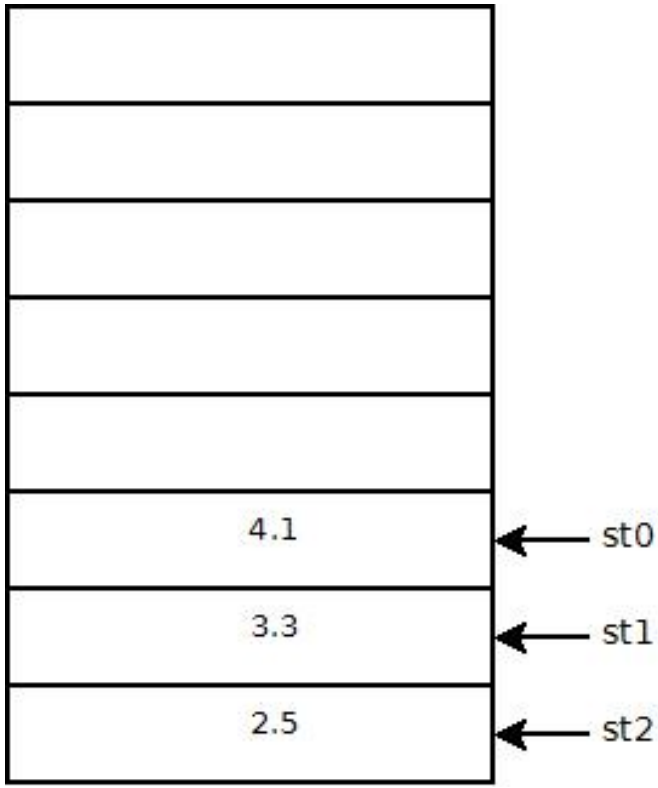
Addition : exemple 2

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



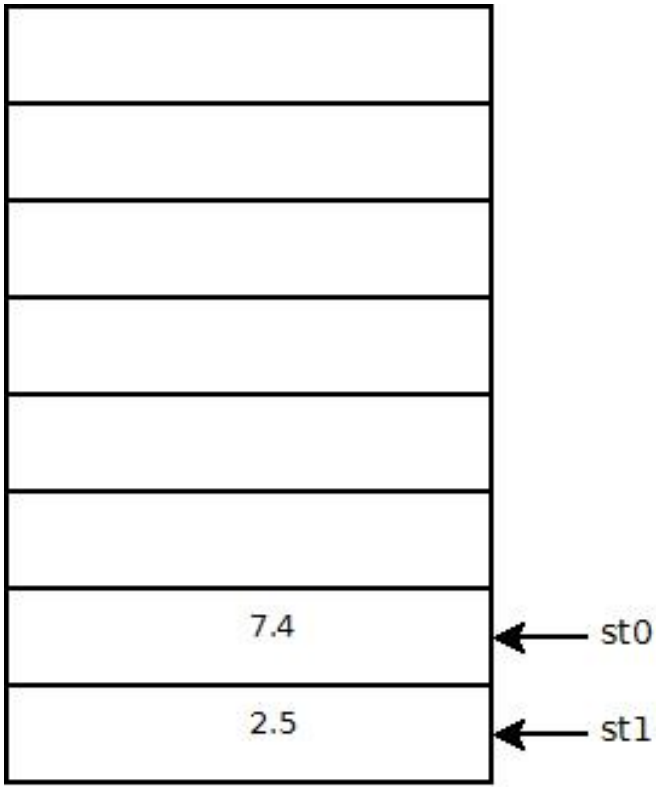
Addition : exemple 2

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



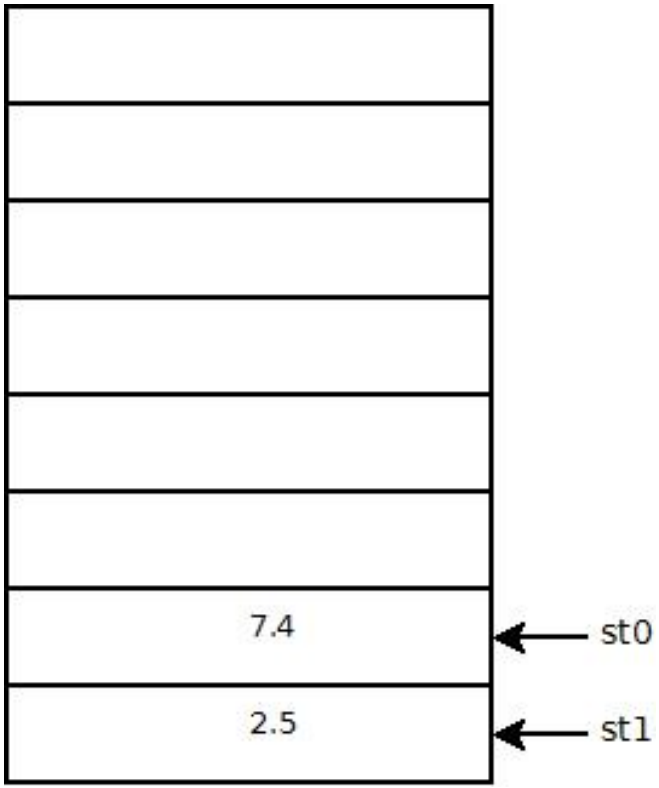
Addition : exemple 2

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



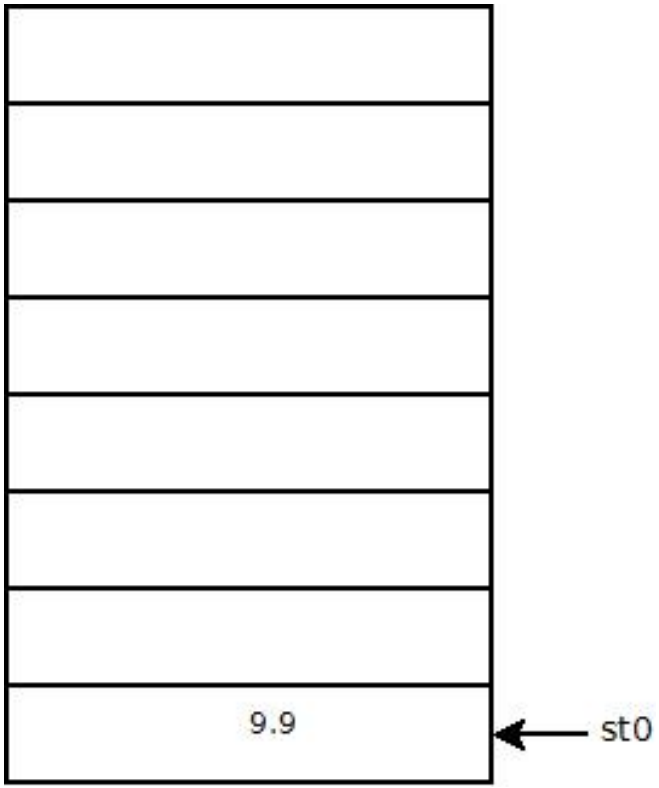
Addition : exemple 2

- $2.5 + 3.3 + 4.1$
- `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

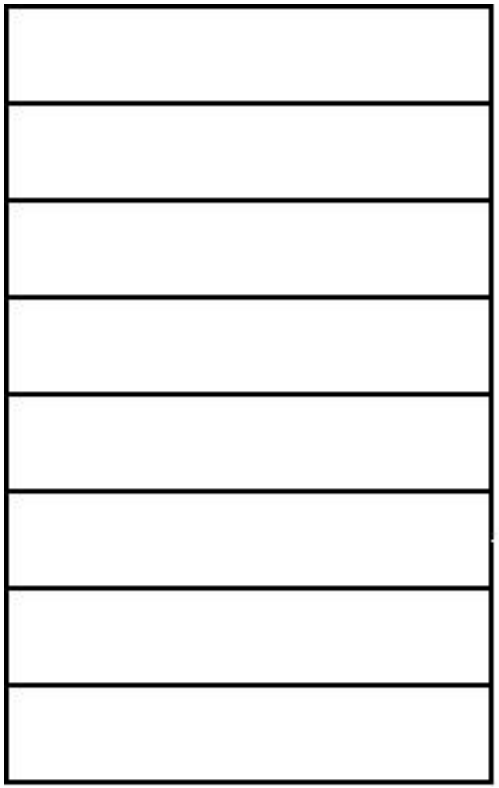
- Soustraction
  - `FSUB/FSUBP` (Floating-point SUBstract Pop)
- Multiplication
  - `FMUL/FMULP` (Floating-point MULtiply Pop)
- Division
  - `FDIV/FDIVP` (Floating-point DIVide Pop)
- Même syntaxe que l'addition (raccourcis inclus)

Exemple

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

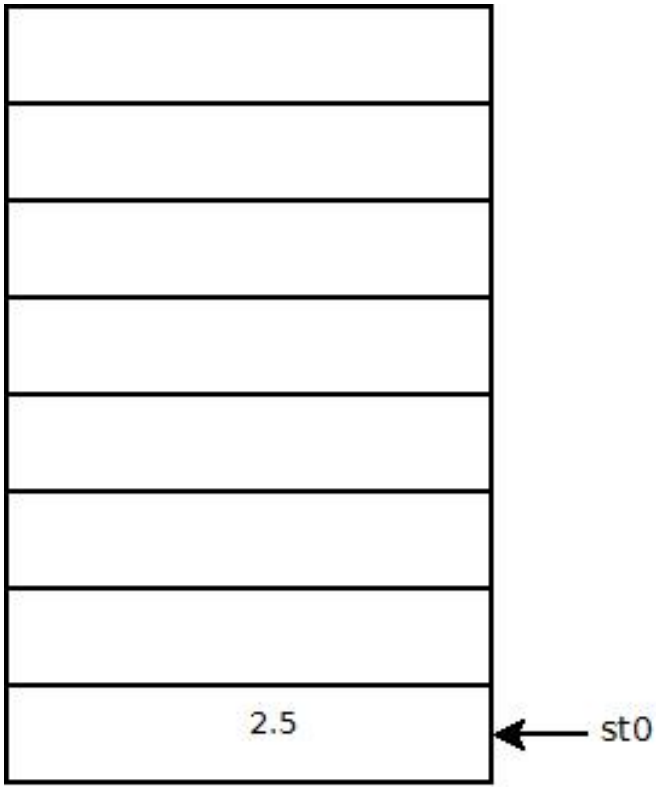
Exemple

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



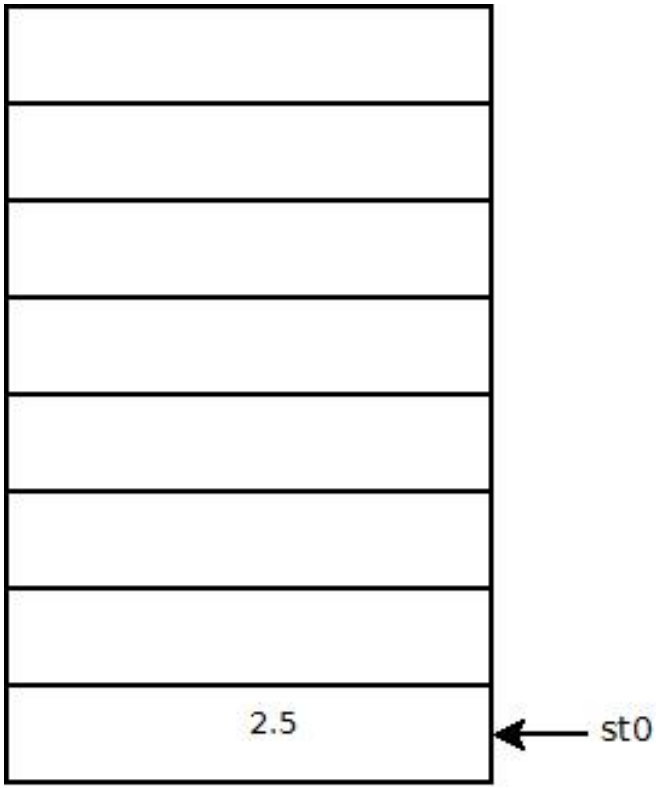
Exemple

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



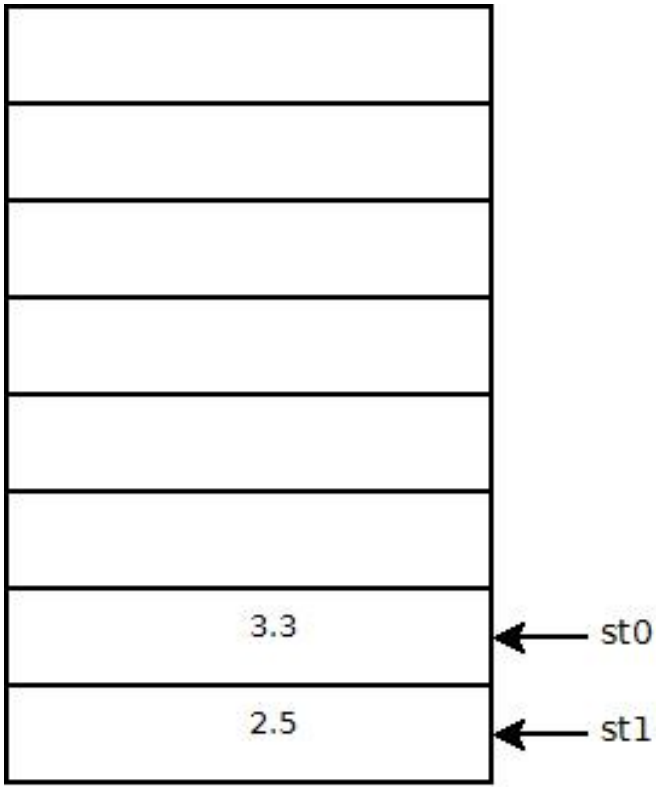
Exemple

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



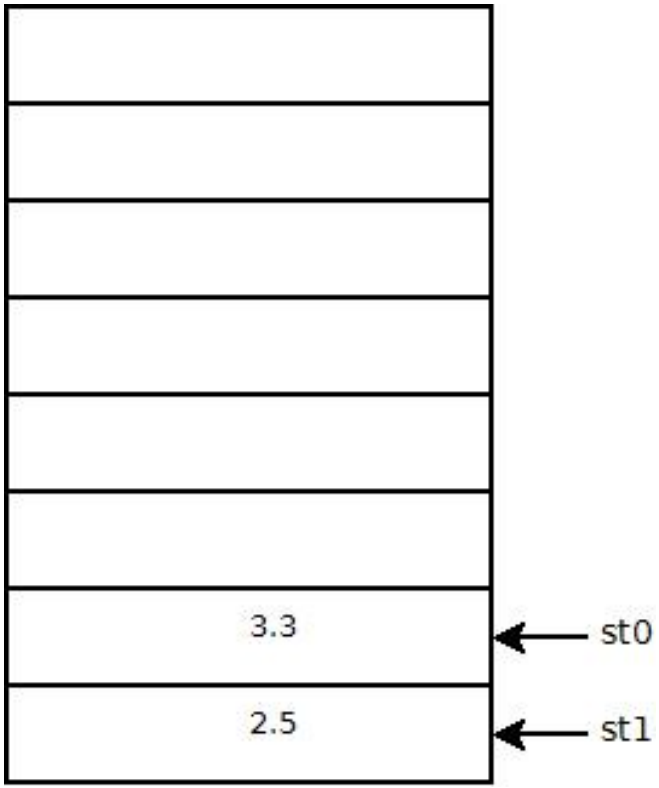
Exemple

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



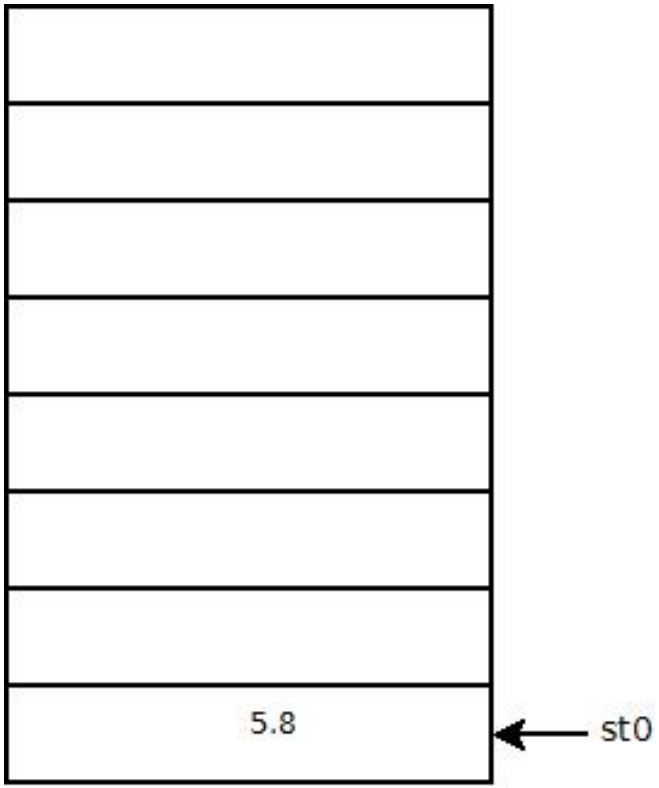
Exemple

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



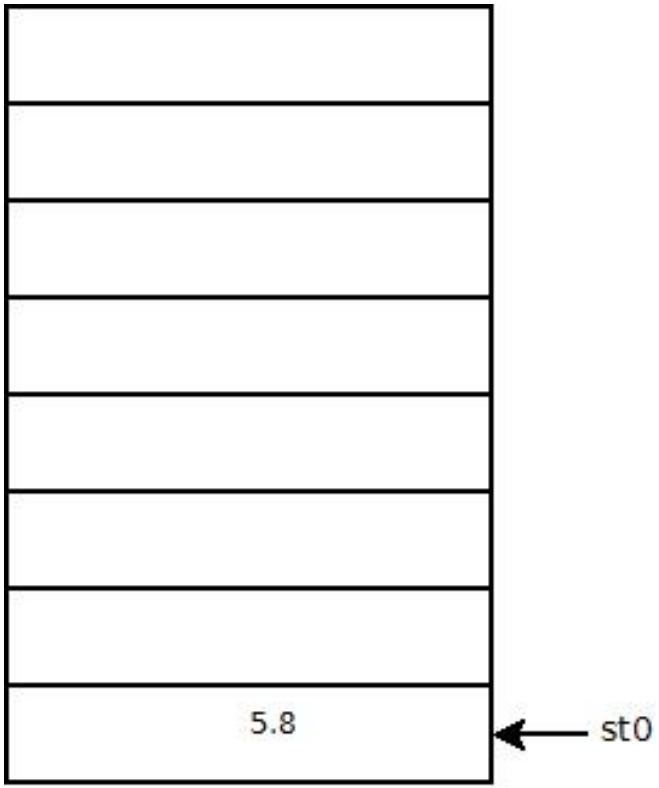
Exemple

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



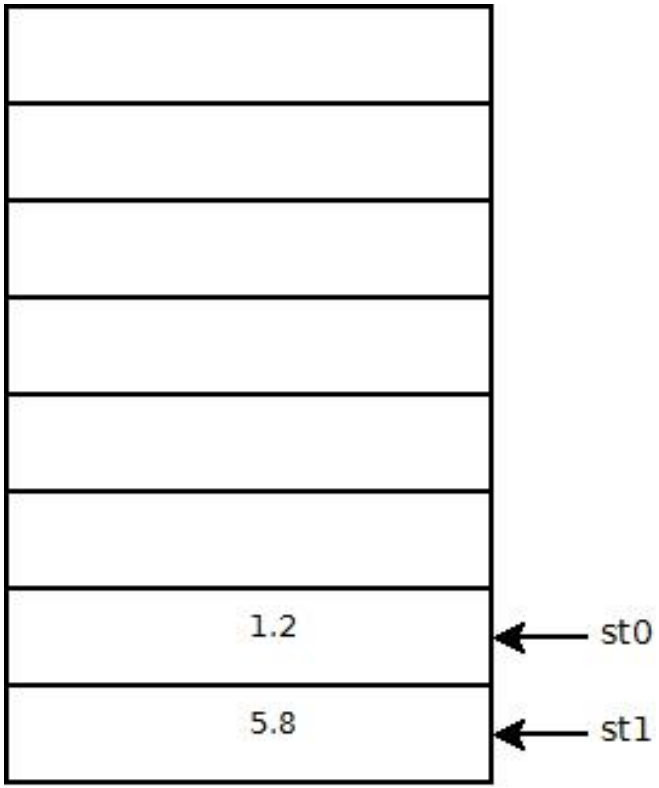
Exemple

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



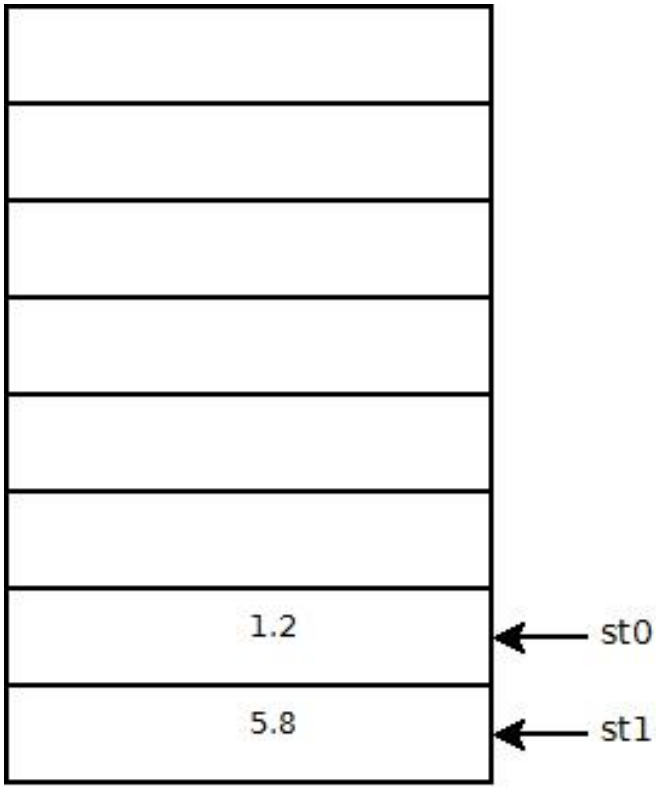
Exemple

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



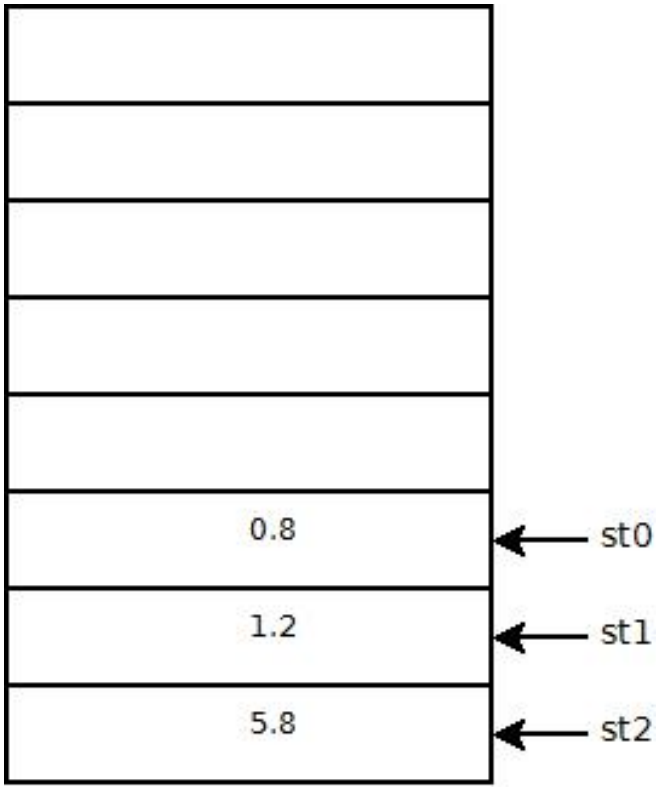
Exemple

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



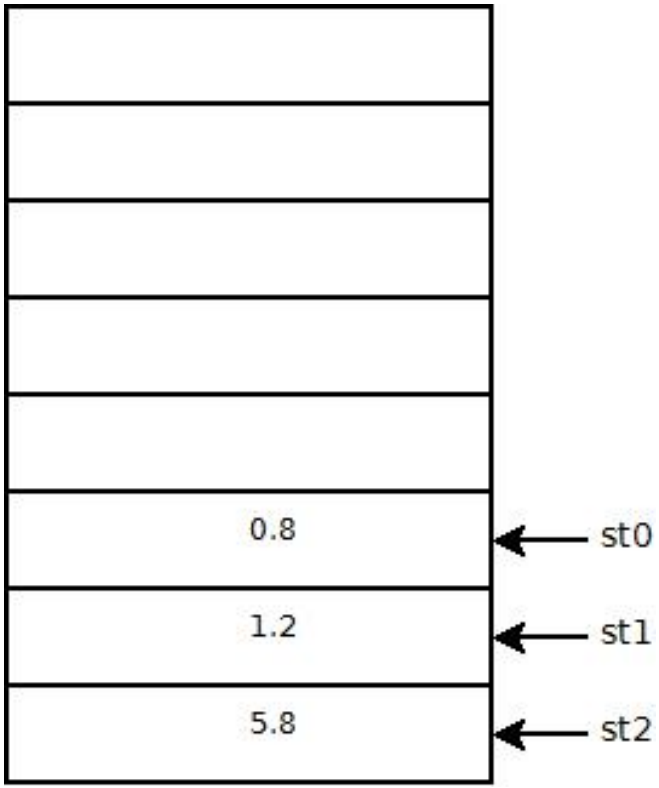
Exemple

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



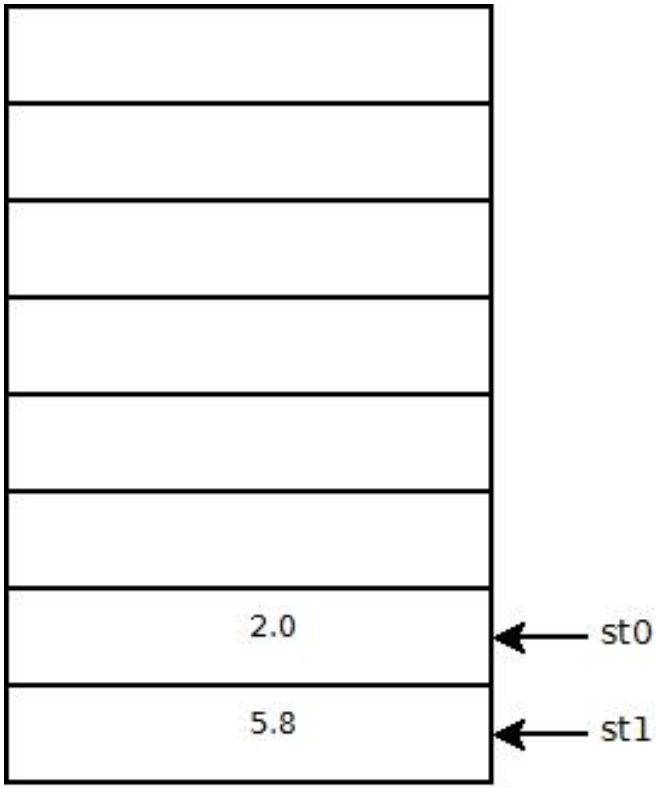
Exemple

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



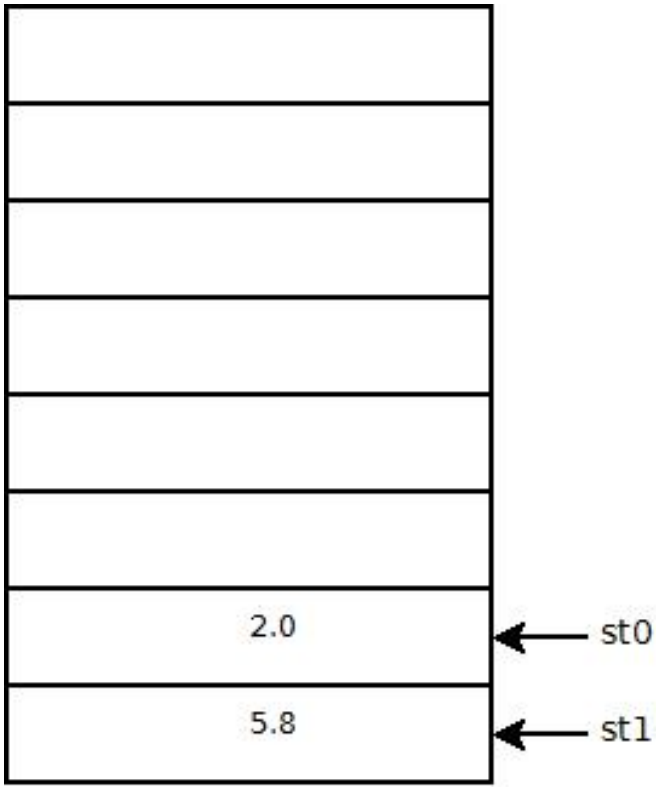
Exemple

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



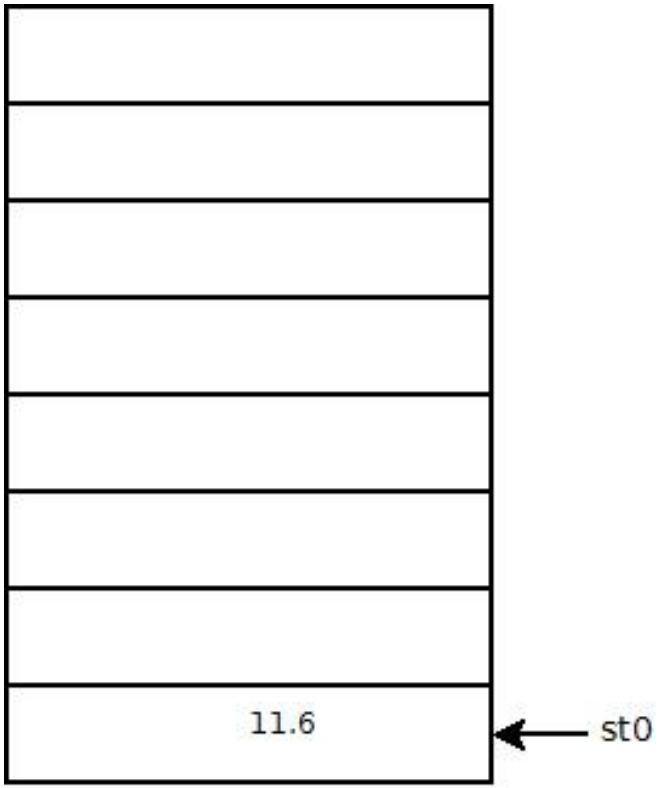
Exemple

- $(2.5 + 3.3) \times (1.2 + 0.8)$
- `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
  - `FABS` = Floating-point ABSolute value
  - `st0` $\leftarrow$ `|st0|`
- Racine carrée
  - `FSQRT` = Floating-point SQuare RooT
  - `st0` $\leftarrow \sqrt{\texttt{st0}}$
- Cosinus
  - `FCOS` = Floating-point COSinus
  - `st0` $\leftarrow$ `cos(st0)`
- Etc...

Conversions réel $\Longleftrightarrow$ entier

- Rajout d'un `I` après le `F` initial (I=Integer)
- Conversion automatique réel $\Longleftrightarrow$ 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
        FMULP

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

        FADDP
        FSQRT
        FISTP dword [dist]
```

Exemple complet

```
        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`