Agenda du cours

- Cours 1 :
  - Généralités archi/assembleur
  - Manipulation émulateur
  - **Code, UAL, registres, mémoire**
  - Exécution, visualisation registres
- Cours 2 : Hiérarchie des mémoires
  - Différents **types de mémoires**
  - Accès mémoire (code, données, E/S)
  - Manipulation structure de données en assembleur
- Cours 3 : Appel de procédures
  - **Notion de Pile**
  - Appel de procédures
  - Passage de paramètres
  - Sauvegarde de contexte d'exécution
- Cours 4 : **Interruptions**
  - Mécanismes internes
  - Programmation d'Interruptions
  - Application aux E/S
- Cours 5 :
  - Développement programme
  - E/S , IT,..
- Cours 6 :
  - **Examen**

Mais avant ….

- Ouvrez vos navigateurs web sur

https://b.socrative.com/student/

- RV dans la salle pour le M211 - Archi C2

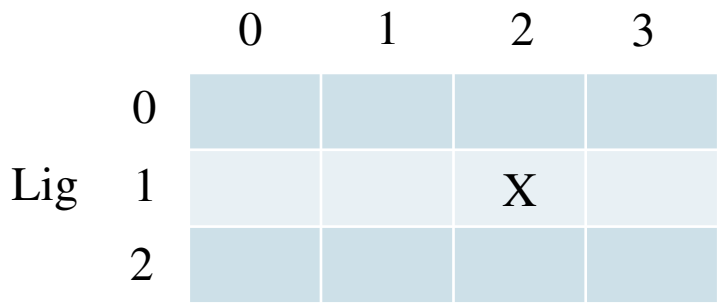
406708

- Mettez votre nom et … répondez aux questions

Exemple d'Adressage

- Soit une matrice 4 colonnes X 3 lignes contenant des octets
- On veut accéder à une case de cette matrice

```
[case] = Offset MAT + (NBCOL * Lig +Col)
```



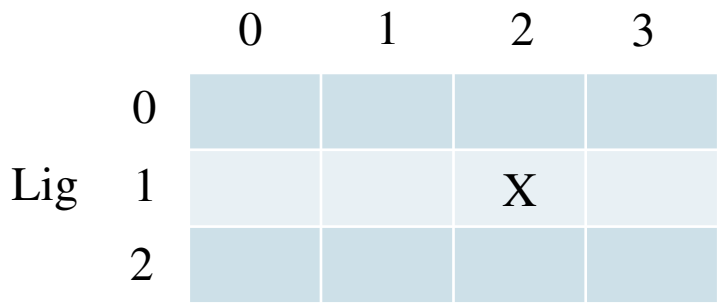
- `[case]=Offset MAT+4*1+2=Offset MAT + 6`
- On utilise un adressage basé indexé pour accéder aux données de cette matrice si les éléments sont sur deux octets.

0

Exemple d'Adressage

- Soit une matrice 4 colonnes X 3 lignes contenant des octets
- On veut accéder à une case de cette matrice

```
[case] = Offset MAT + (NBCOL * Lig +Col)
```



- `[case]=Offset MAT+4*1+2=Offset MAT + 6`
- On utilise un adressage basé indexé pour accéder aux données de cette matrice si les éléments sont sur deux octets.

0

Utilité de la Pile

- La pile est une zone mémoire particulière
- Elle offre un moyen d'accéder à des données en mémoire principale et est très utilisée pour stocker temporairement des valeurs.
- Elle mémorise les adresses d'appel et de retour de sous programmes
- Elle mémorise les contextes d'exécution

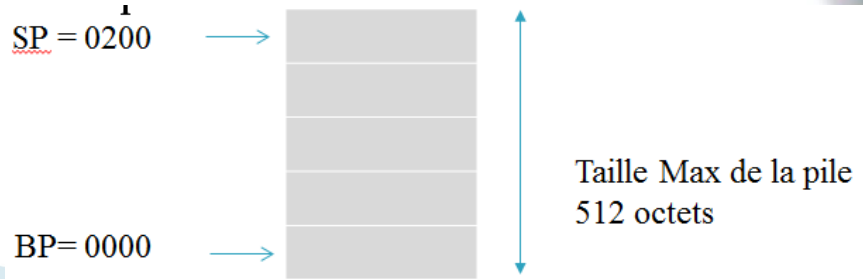
Registres liés à la Pile

- Zone de rangement de données
  - Zone mémoire spécialisée
  - Fonctionnement LIFO Last In First Out

- 3 registres pour repérer la zone mémoire de Pile
  - Sommet de Pile: SP (Stack Pointeur)
  - Base de la Pile: BP, (Base Pointeur)
  - Dans le segment SS (Stack Pointeur)

- Expl de réservation de la pile :

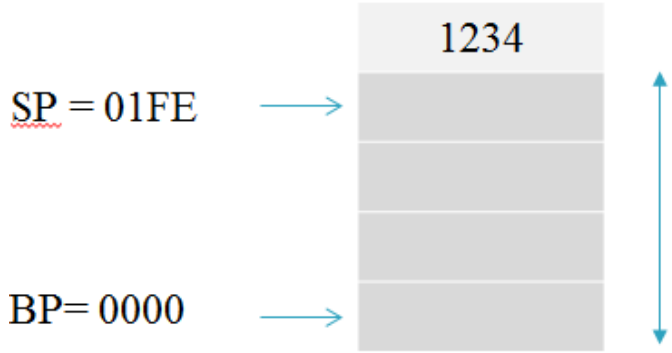
```
PILE SEGMENT STACK
DW 256 DUP(?)
Base:
PILE ENDS
```



Manipulation de la Pile

- Instructions pour accéder à la pile
  - PUSH <*registre*> : Empile le contenu du registre *sur le sommet de la pile*
  - POP <*registre*> : dépile le contenu du sommet de la pile dans le registre

- Impact sur les registres
  - SP <- SP – taille du registre
  - BP identique

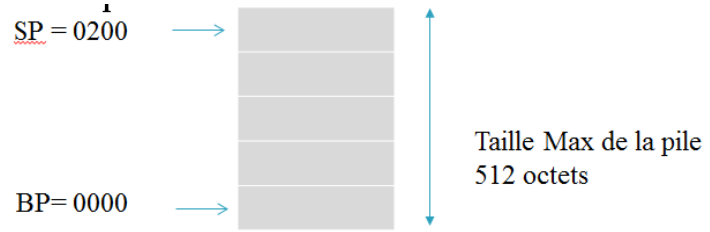


Zone disponible
restante
510 octets

Registres liés à la Pile

- Expl de réservation de la pile:

```
PILE SEGMENT STACK
DW 256 DUP(?)
Base:
PILE ENDS
```

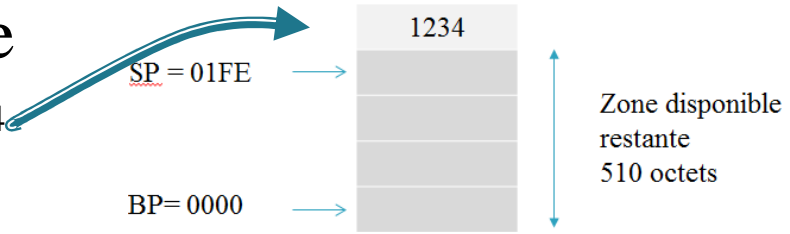


- Impact des instructions sur la pile

```
MOV AX, 1234
PUSH AX
```

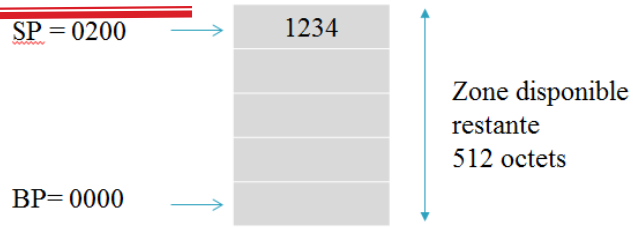
AX = XXXX → AX = 1234

BX = XXXX




```
POP BX
```

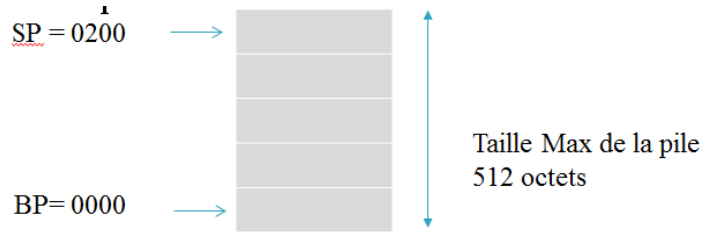
BX = 1234



Exemple d'évolution de la pile d'à la Pile

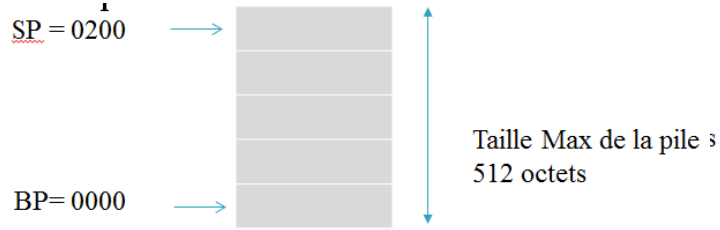
- Expl de réservation de la pile:

```
PILE SEGMENT STACK
DW 256 DUP(?)
Base:
PILE ENDS
```



- Exemple de code qui modifie la pile

	AX = XXXX	BX = YYYY
MOV AX, 250	AX = 00FA	BX = YYYY
PUSH AX		
SUB AX 100	AX = 0096	BX = YYYY
PUSH AX		
POP BX	AX = 0096	BX = 0096
SUB AX, BX	AX = 0000	BX = 0096
POP AX	AX = 00FA	BX = 0096



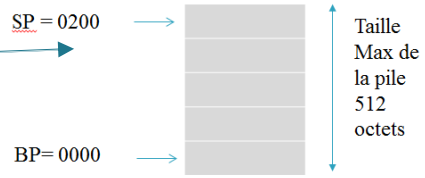
Rôle de la pile dans l'appel de sous programme

- Un **sous-programme** est un bout de code qui
  - est appelé par un programme principal
  - Se situe en mémoire à une adresse qui est souvent dans une autre zone de code que le programme appelant

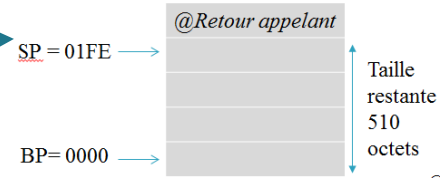
```
PILE SEGMENT STACK
    DW 256 DUP(?)
    Base:
PILE ENDS

DATA     SEGMENT
ValX        dw      ?

DATA     ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA,SS:PILE


main PROC
    MOV AX,DATA     ;Positionnement de DS
    MOV DS,AX       ;Dans le segment de donnees

    MOV AX,PILE     ;Positionnement de DS
    MOV SS,AX       ;Dans le segment de PILE

    ; appel du sous programme
    CALL  SousProgramme

    MOV AH,4CH
    INT 21H
main ENDP

SousProgramme   PROC
    MOV AX, 1111h
    MOV ValX, AX
    ret
SousProgramme ENDP

CODE ENDS
```

Rôle de la pile dans l'appel de sous programme

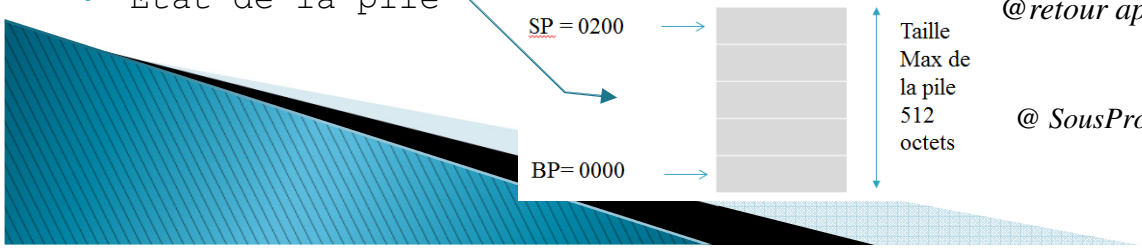
- Un **sous-programme** est un bout de code qui
  - est appelé par un programme principal
  - Se situe en mémoire à une adresse qui est souvent dans une autre zone de code que le programme appelant
- Avant appel
  - CS = Segment code
  - IP = @CALL SousProg
  - Etat de la pile



- call SousProgramme
  - CS = Segment code
  - IP = @ SousProg
  - Etat de la pile



- ret
  - CS = Segment code
  - IP = @ retourappelant
  - Etat de la pile




```
PILE SEGMENT STACK
     DW 256 DUP(?)
     Base:
PILE ENDS

DATA      SEGMENT
ValX        dw      ?

DATA      ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA,SS:PILE


main PROC
     MOV AX,DATA      ;Positionnement de DS
     MOV DS,AX        ;Dans le segment de donnees

     MOV AX,PILE      ;Positionnement de DS
     MOV SS,AX        ;Dans le segment de PILE

     ; appel du sous programme
     CALL  SousProgramme

     MOV AH,4CH
     INT 21H
main ENDP

SousProgramme    PROC
     MOV AX, 1111h
     MOV ValX, AX
     ret
SousProgramme ENDP

CODE ENDS
```

@retour appelant

@ SousProg

Rôle de la pile dans l'appel de sous programme

- **Point d'entrée** ou @sous programme adresse de la première instruction exécutée dans le SP
- **L'appel d'un sous-programme** : lors d'un appel on doit sauvegarder l'adresse de retour du SP c-à-d
  - CS : lorsque le S-P est extra segment
  - IP : dans tous les cas

  NB: Les sauvegardes se font sur la pile.
- **Instructions de sauvegarde** : si le S-P utilise des registres du processeur, il est important se sauvegarder ses registres sur la pile au début du SP.
- **Corps du programme** : le S-P réalise des opérations sur les registres du processeur. Le S-P utilise les ressources de la machine.

Rôle de la pile dans le passage de paramètres

- **Un sous programme**
  - Effectue un traitement sur des données (paramètres) transmis par le programme appelant.
  - Produit un résultat transmis au programme appelant
- 2 méthodes de passage de paramètres
  - Par registres
    - paramètres déposés avant l'appel
    - Paramètres de sortie rangés dans des registres
  - Par la pile
    - paramètres empilés avant l'appel par le PP
    - Résultats de sortie rangés dans la pile par le SP
    - Résultats récupérés sur la pile par le PP

Rôle de la pile dans le passage de paramètres

- **Exemple passage par registres**
  - Programme principal

```
; passage des parametres par registres
MOV AX, Var1
MOV BX, Var2
; appel du sous programme
CALL   Minimum
; resultat dans AX
MOV VarMin, AX
MOV AH,4CH
INT 21H
```

Sous programme

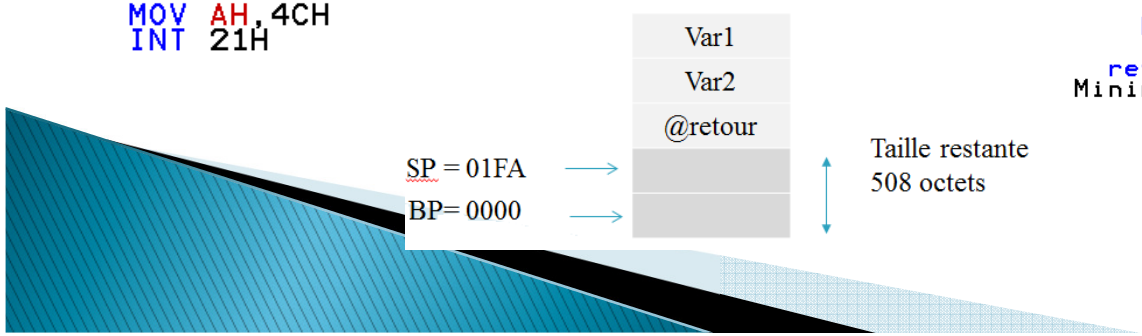
```
Minimum PROC
    CMP AX, BX
    JLE leMin
        MOV AX,BX
    leMin:
  ret
Minimum ENDP
```

- **Exemple passage par la pile**
  - Programme principal

```
; passage des parametres par la pile
PUSH Var1
PUSH Var2
; appel du sous programme
CALL   Minimum
; resultat dans AX
POP AX

MOV AH,4CH
INT 21H
```

Sous programme

```
Minimum PROC
    ; recuperation des parametres sur la pile.
    MOV BP, SP
    MOV AX, [BP+4]
    MOV BX,[BP+2]
    CMP AX, BX
    JLE leMin
        MOV AX,BX
    leMin:
    ; depot du resultat sur la pile
    MOV  [BP+4], AX

  ret
Minimum ENDP
```



Rôle de la pile sauvegarde du contexte d'éxécution

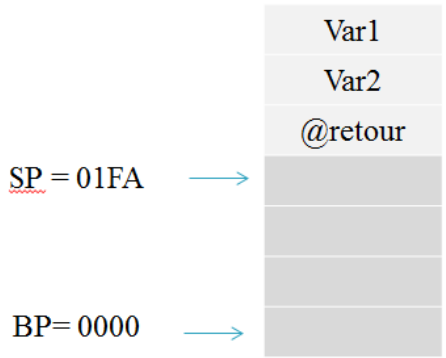
- Programme principal

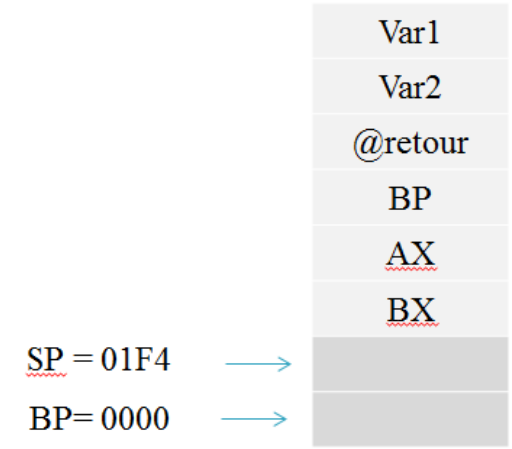
```
; passage des parametres par la pile
PUSH Var1
PUSH Var2
; appel du sous programme
CALL  Minimum
; resultat dans AX
POP VarMin
```

- Sous programme

```
Minimum PROC
    ; sauvegarde du contexte
    PUSH BP
    PUSH AX
    PUSH BX
    ; recuperation des parametres sur la pile.
    MOV BP, SP
    MOV AX, [BP+10]
    MOV BX,[BP+8]
    CMP AX, BX
    JLE leMin
        MOV AX,BX
    leMin:
    ; depot du resultat sur la pile
    MOV  [BP+10], AX
    ; restauration du contexte
    POP BP
    POP AX
    POP BX

  ret
Minimum ENDP
```

- Etat de la Pile

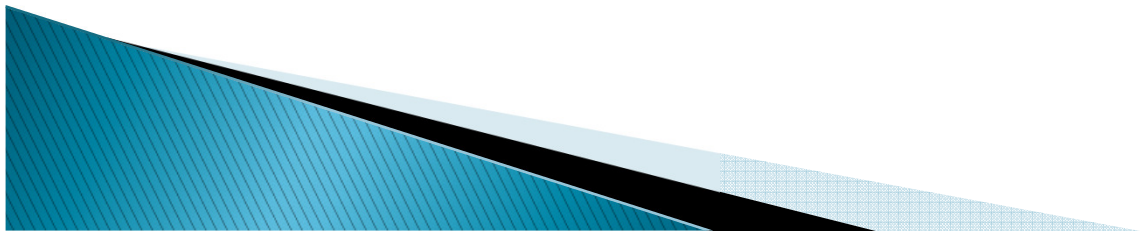




Rôle du compilateur

- Le rôle d'un compilateur vers assembleur est de réaliser ce travail de
  - Passage de paramètres: cjoix par registre ou par pile
  - Sauvegarde du contexte systématique de l'ensemble des registres

- Optimisation du passage des paramètres et de la sauvegarde des registres



Rôle du compilateur

- Programme principal

```
; passage des parametres par la pile
PUSH Var1
PUSH Var2
; appel du sous programme
CALL  Minimum
; resultat dans AX
POP VarMin
```

Sous programme

```
Minimum PROC
     ; sauvegarde du contexte
     PUSH BP
     PUSH AX
     PUSH BX
     ; recuperation des parametres sur la pile.
     MOV BP, SP
     MOV AX, [BP+10]
     MOV BX,[BP+8]
     CMP AX, BX
     JLE leMin
          MOV AX,BX
     leMin:
     ; depot du resultat sur la pile
     MOV  [BP+10], AX
     ; restauration du contexte
     POP BP
     POP AX
     POP BX

  ret
Minimum ENDP
```

- Etat de la Pile