Architecture des Ordinateurs et Systèmes d'Exploitation

Cours n°5

Le langage assembleur (2): Pile, Procédures.
Les Interruptions

Ph. Leray

3ème année

L'assembleur 8086 : la pile

= structure de « rangement » de données

- **définie dans un segment de mémoire particulier**
- **fonctionnement *LIFO* (Last In, First Out)** **ex: pile d'assiettes**
- **comment y accéder :**
  - **adresse du dernier élement posé : SS:SP**
  - **empiler : PUSH Reg16 (registre 16 bits)**
  - **dépiler : POP Reg16 (registre 16 bits)**
- **« la pile augmente vers les adresses faibles »**

  (PUSH : SP←SP-2, POP : SP←SP+2)
- **Déclaration du segment de pile :**

Déclaration d'une pile de 1024 octets (=1 Ko) initialisés à 0

```
Assume SS:Pile

Pile SEGMENT STACK
                dw 1024 dup (0)
Pile ENDS
```

L'assembleur 8086 : les procédures (1/5)

- **Déclaration d'une procédure :**
  - **appel proche (near) : la procédure et le programme sont dans le même segment de code**
  - **appel lointain (far) : la procédure et le programme sont dans deux segments de code différents**

```
Moyenne PROC [NEAR/FAR]
        instructions
Moyenne ENDP
```

- **Appel de la procédure dans le programme**

```
Call Moyenne
```

Comment l'unité de traitement arrive-t-elle à retourner au programme principal à la fin de la fonction ?

- **Au moment de l'appel de la fonction, l'adresse de l'instruction suivante est sauvegardée dans la pile :**
  - **appel proche (near) : sauvegarde de IP**
  - **appel lointain (far) : sauvegarde de CS puis de IP**

⇒ ***A la fin de la procédure, l 'UT récupère les valeurs sauvegardées pour retourner au programme principal***

L'assembleur 8086 : les procédures (2/5)

- **2 méthodes de passage de paramètres : * par registre * par la pile**

- **Passage de paramètres par registre :**
  - **les paramètres d'entrée de la procédure sont mis dans des registres avant l 'appel de la procédure**
  - **les paramètres de sortie sont aussi rangés dans des registres**
  - **avantage = rapidité inconvénients = peu de registres**
    pas de « portabilité »
  - **Ex:**

```
; programme principal
            ...
            Mov AX, N1
            Mov BX, N2
            Call Moyenne
            Mov Res, AX
            ...
```

```
Moyenne PROC
            Add AX, BX
            SHR AX, 1
            Ret
Moyenne ENDP
```

Ret = dépilement de IP pour retourner au programme principal

L'assembleur 8086 : les procédures (3/5)

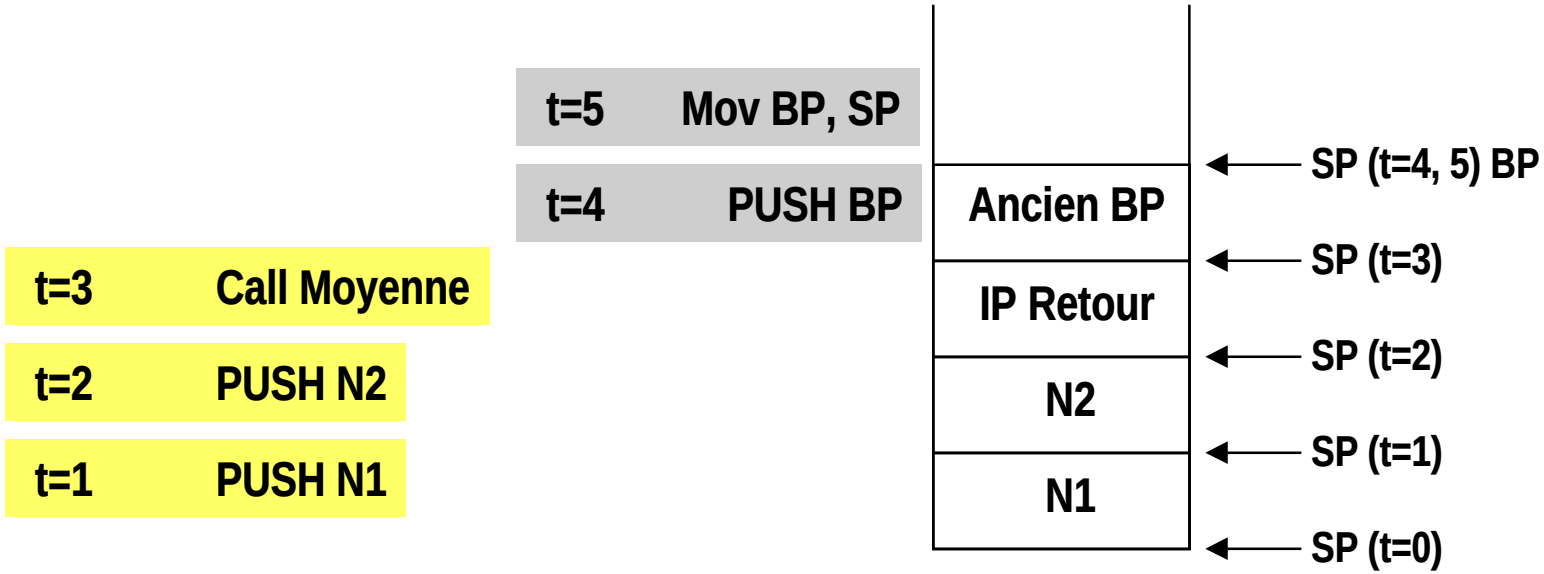
- **Passage de paramètres par la pile :**
  - **les paramètres d'entrée sont empilés avant l 'appel de la procédure**
  - **les paramètres de sortie sont dépilés par le programme principal**
  - **avantage = « portabilité »** **inconvénients = récupération des paramètres plus « lourde »**
  - **Ex:**

```
; programme principal
        ...
        PUSH N1
        PUSH N2
        Call Moyenne
        POP Res
        ...
```

```
Moyenne PROC
        PUSH BP
        Mov BP, SP
        Mov AX, [BP+4]
        Add AX, [BP+6]
        SHR AX, 1
        Mov [BP+6], AX
        POP BP
        Ret 2
Moyenne ENDPc
```

L'assembleur 8086 : les procédures (4/5)

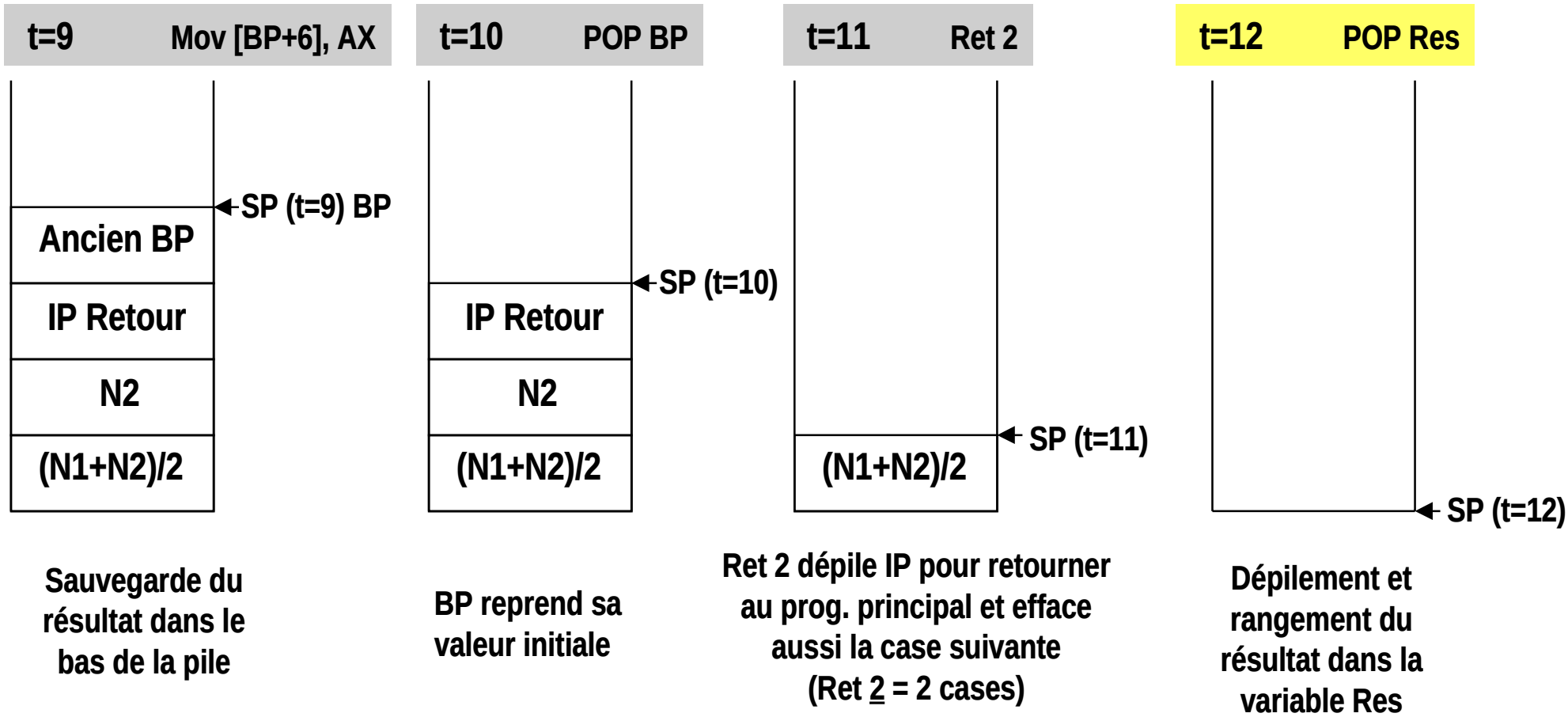
- **Passage de paramètres par la pile : évolution de la pile**



- **A quoi sert BP ?**
  - **BP = référence de la pile au début de la procédure**
  - **permet d'avoir une référence fixe pour récupérer les paramètres**
    ici : [BP+4] pour N2 et [BP+6] pour N1

L'assembleur 8086 : les procédures (5/5)

– **Sauvegarde du résultat et « nettoyage de la pile » :**



Sauvegarde du résultat dans le bas de la pile

BP reprend sa valeur initiale

Ret 2 dépile IP pour retourner au prog. principal et efface aussi la case suivante (Ret 2 = 2 cases)

Dépilement et rangement du résultat dans la variable Res

Quels paramètres ?

- **On peut utiliser deux paramètres différents :**
  - **la valeur de la variable**
  - **l 'adresse de la variable**

```
; passage des valeurs par registre
        ...
        Mov AX, N1
        Mov BX, N2
        Call Moyenne_par_valeur
        Mov Res, AX
        ...
```

```
Moyenne_par_valeur PROC
        Add AX, BX
        SHR AX, 1
        Ret
Moyenne_par_valeur ENDP
```

```
; passage des adresses par registre
        ...
        Mov SI, Offset N1
        Mov DI, Offset N2
        Mov BX, Offset Res
        Call Moyenne_par_adresse
        ...
```

```
Moyenne_par_adresse PROC
        Mov AX, [SI]
        Add AX, [DI]
        SHR AX, 1
        Mov [BX],AX
        Ret
Moyenne_par_adresse ENDP
```

(Idem pour le passage par pile)

Procédure sale / procédure propre

Procédure sale = modifie les registres

```
Moyenne PROC
        PUSH BP
        Mov BP, SP
        Mov AX, [BP+4]
        Add AX, [BP+6]
        SHR AX, 1
        Mov [BP+6], AX
        POP BP
        Ret 2
Moyenne ENDP
```

A la fin de la procédure, AX est modifié

Procédure propre = ne modifie pas (ou restaure en fin de procédure) les registres

```
Moyenne PROC
        PUSH BP
        Mov BP, SP
        PUSH AX
        Mov AX, [BP+4]
        Add AX, [BP+6]
        SHR AX, 1
        Mov [BP+6], AX
        POP AX
        POP BP
        Ret 2
Moyenne ENDP
```

AX est sauvegardé en début de procédure et restauré à la fin

Les interruptions

- **En général, l 'Unité de Traitement exécute séquentiellement les instructions ou effectue des sauts programmés (JMP, CALL)**

- **Il existe des situations où l 'U.T. est « déroutée » de sa tâche :**
  - **Reset : signal envoyé au processeur pour un (re-)démarrage**
  - **Exceptions (interruptions internes) : débordement de pile, dépassement de capacité, div /0, …**
  - **Appels systèmes (int. logicielles) : appels du programme lui-même « int 21h » …**
  - **Interruptions physiques (int. externes) : appels d'autres périphériques**

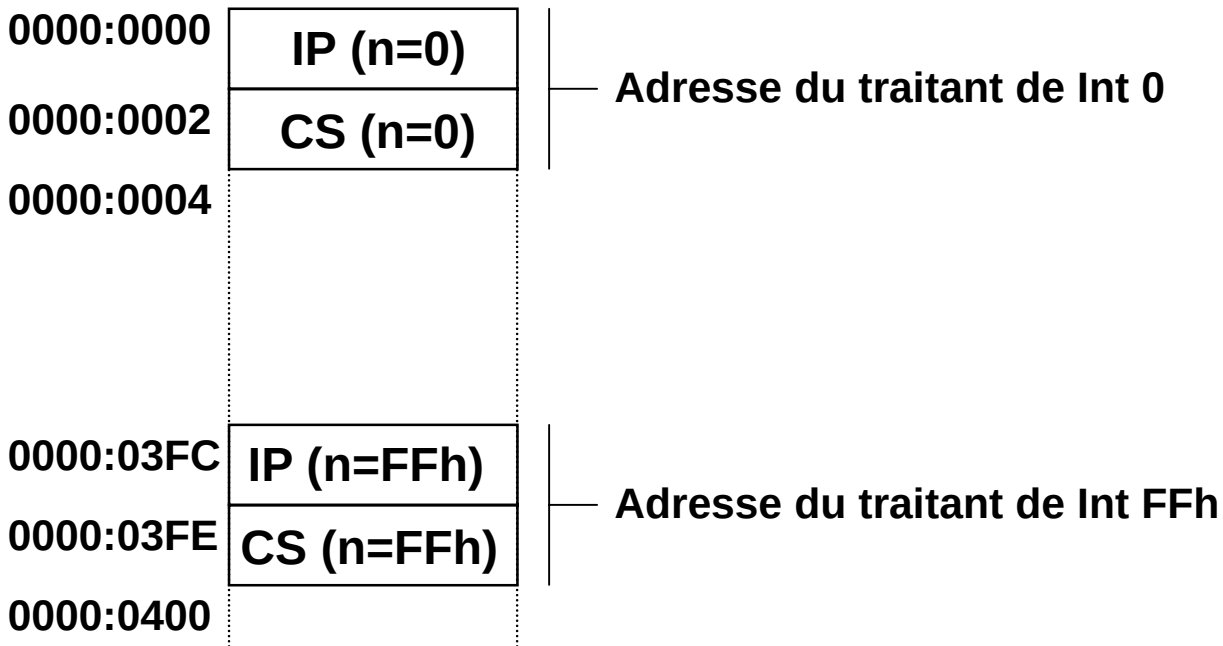
- **Les interruptions sont +/- prioritaires**
  - **Int. non masquables : l 'UT est obligée de s 'en occuper (Reset,…)**
  - **Int. masquables : l 'UT peut ne pas en tenir compte (grâce au drapeau IF)**

Les interruptions : fonctionnement 1/2

- **Chaque interruption possède un numéro de 0 à 255 (FFh) (ex: Int 21h)**
- **Lorsque l'interruption est déclenchée, l 'U.T. doit exécuter un bout de programme bien précis : le traitant de l 'interruption.**

⇒ **l 'U.T. doit savoir où se trouve l 'adresse *(CS:IP)* du traitant numéro *n***

– **Cette information est stockée en mémoire (0000:0000 à 0000:03FFh)**



Les interruptions : fonctionnement 2/2

- **Au démarrage, la table des interruptions contient des valeurs « par défaut »**
- **Il est possible de modifier cette table :**
  - **pour ajouter/supprimer un traitant**
  - **surcharger un traitant (ex: ajouter un message d 'erreur en plus du div/0)**

- **Comment ? : appels systèmes (ici l 'exemple du DOS)**
  - **Lecture de l 'adresse du traitant nn :**

    Fonction DOS 35h (entrée : AH=35h, AL=nn sortie : ES=CS(nn), BX=IP(nn))

  - **Modification de l 'adresse du traitant nn :**

    Fonction DOS 25h (entrée : AH=25h, AL=nn, DS=CS(nn), DX=IP(nn))

  - **Retour au dos en laissant le traitant en mémoire (TSR = Terminate and Stay Resident)**

    Fonction DOS 31h (entrée : AH=31h, AL=nn, DX=Taille du traitant en blocs de 16 octets)