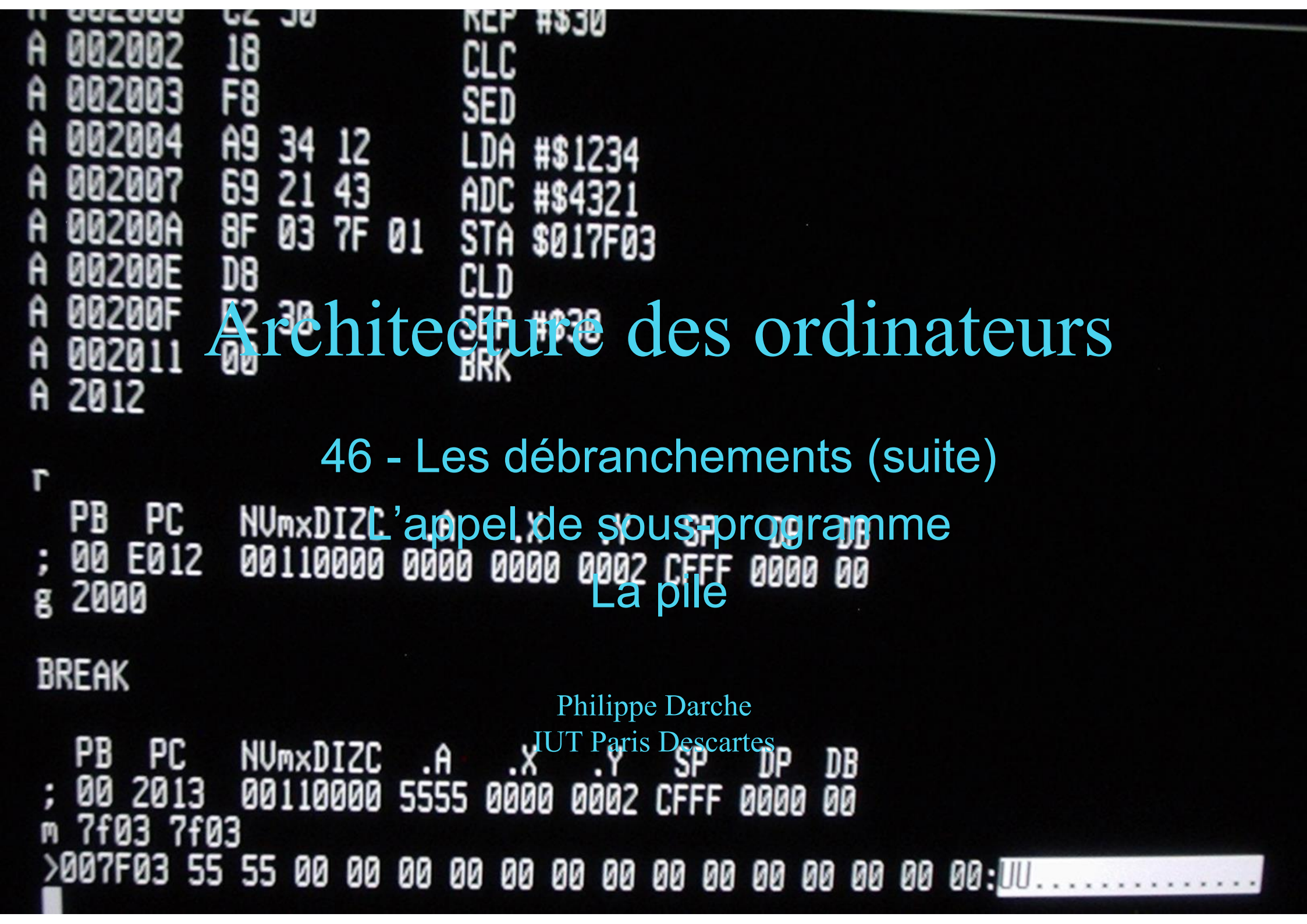


Architecture des ordinateurs

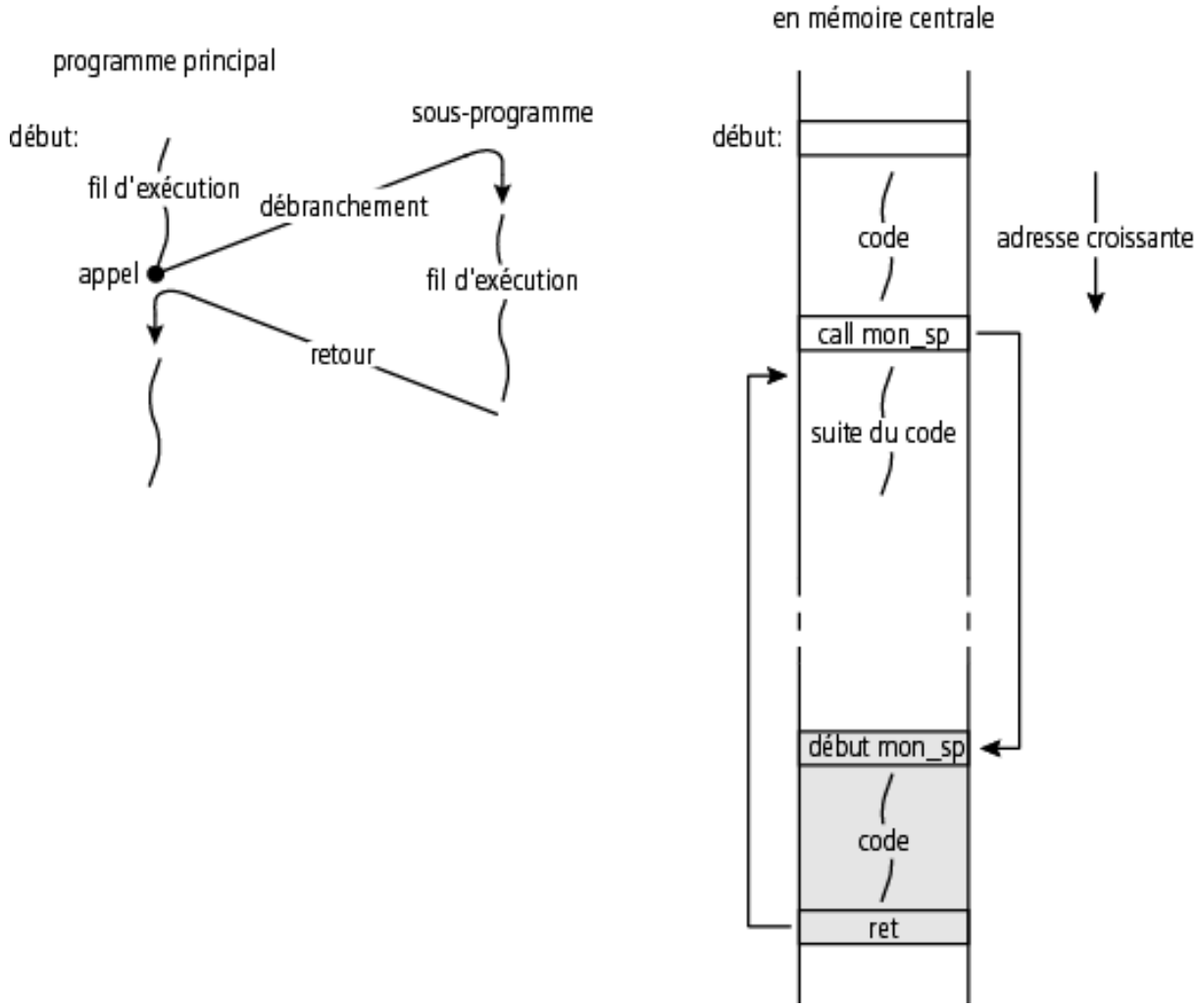
46 - Les débranchements (suite)

L'appel de sous-programme

La pile

Philippe Darche
IUT Paris Descartes

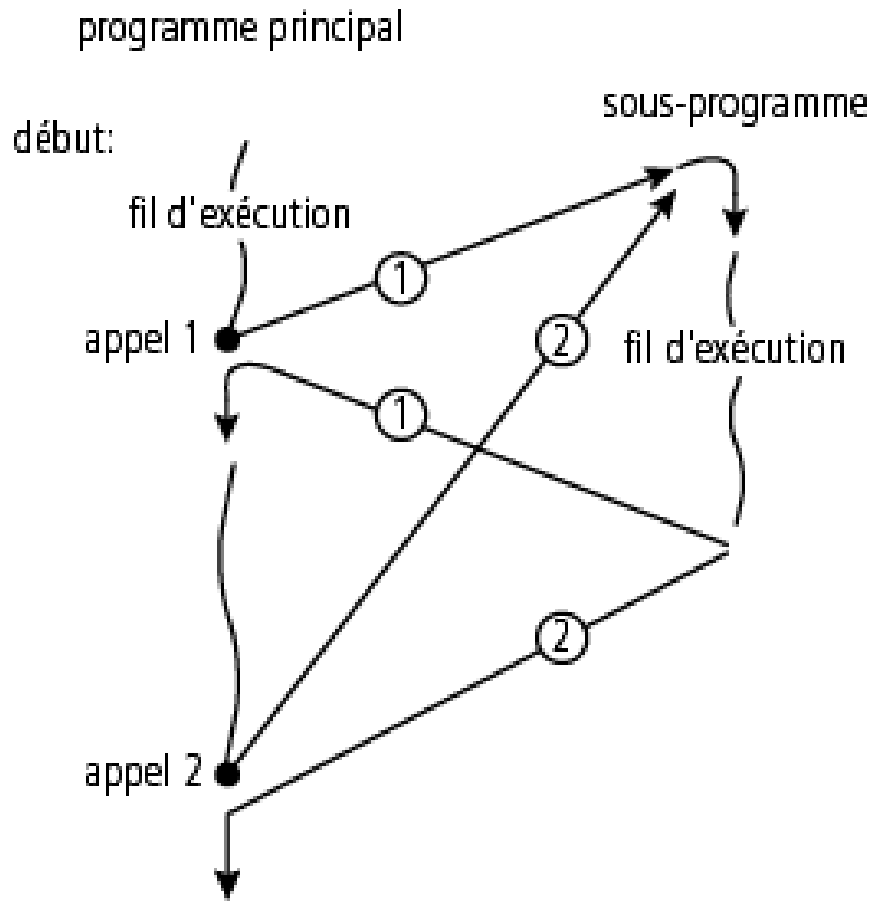
Le problème


programme principal
sous-programme
début:
fil d'exécution
débranchement
appel
fil d'exécution
retour
en mémoire centrale
début:
code
adresse croissante
call mon_sp
suite du code
début mon_sp
code
ret

Le sous-programme

- Assimilable à la fonction du langage C
- Utilité : factorisation du code
- Instructions du langage d'assemblage associées
  - appel : call
  - retour : ret

Comment résoudre les adresses de saut et de retour ?



Comment résoudre les adresses de saut et de retour ?

- Le saut
  - l'adresse contenu dans IP est sauvegardé dans la pile
    - le contexte minimum
  - le registre IP va recevoir l'adresse associée au *call*
- Le retour
  - le registre IP reçoit l'adresse de retour

La pile

- Structure de données de type LIFO (*Last In First Out*)
  - la donnée la dernière entrée est la première sortie
- Deux opérations d'accès de base
  - empiler() et dépiler()
    - chez Intel, instructions push et pop (voir exemples d'utilisation plus loin)
- Etat : vide
- Information : nombre d'éléments à un instant t

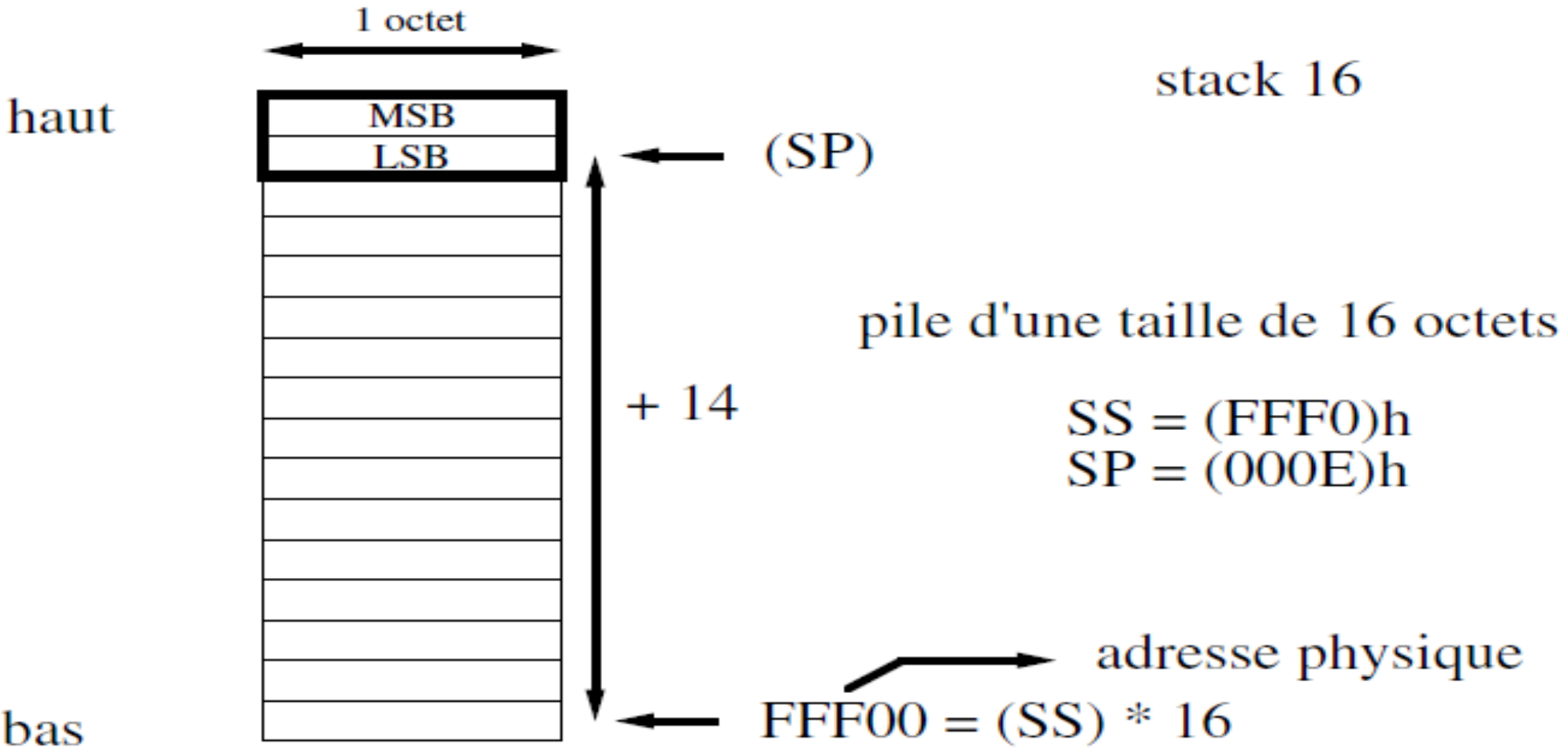
La pile (suite)

- Risque : le dépassement ou débordement de pile (*stack overflow*)
  - dépassement de l'espace alloué
- Des machines n'utilisent que la pile comme espace mémoire à la place des registres
  - *stack computer*

La pile (*stack*) dans le 80x86

- Implémentée chez Intel sous la forme d'un segment (*i.e.* zone d'octets contigus)
- Taille de la pile = taille d'un segment
  = 64 Kio maximum
- Registres de gestion :
  - SS : *Stack Segment*
    - le registre SS pointe le début du segment de pile en mémoire centrale
  - SP : *Stack Pointer*
    - le registre SP pointe la dernière donnée empilée = haut de la pile (TOS : *Top Of Stack*)
  - BP : *Base Pointer*
    - le registre BP est en général utilisé pour récupérer des paramètres

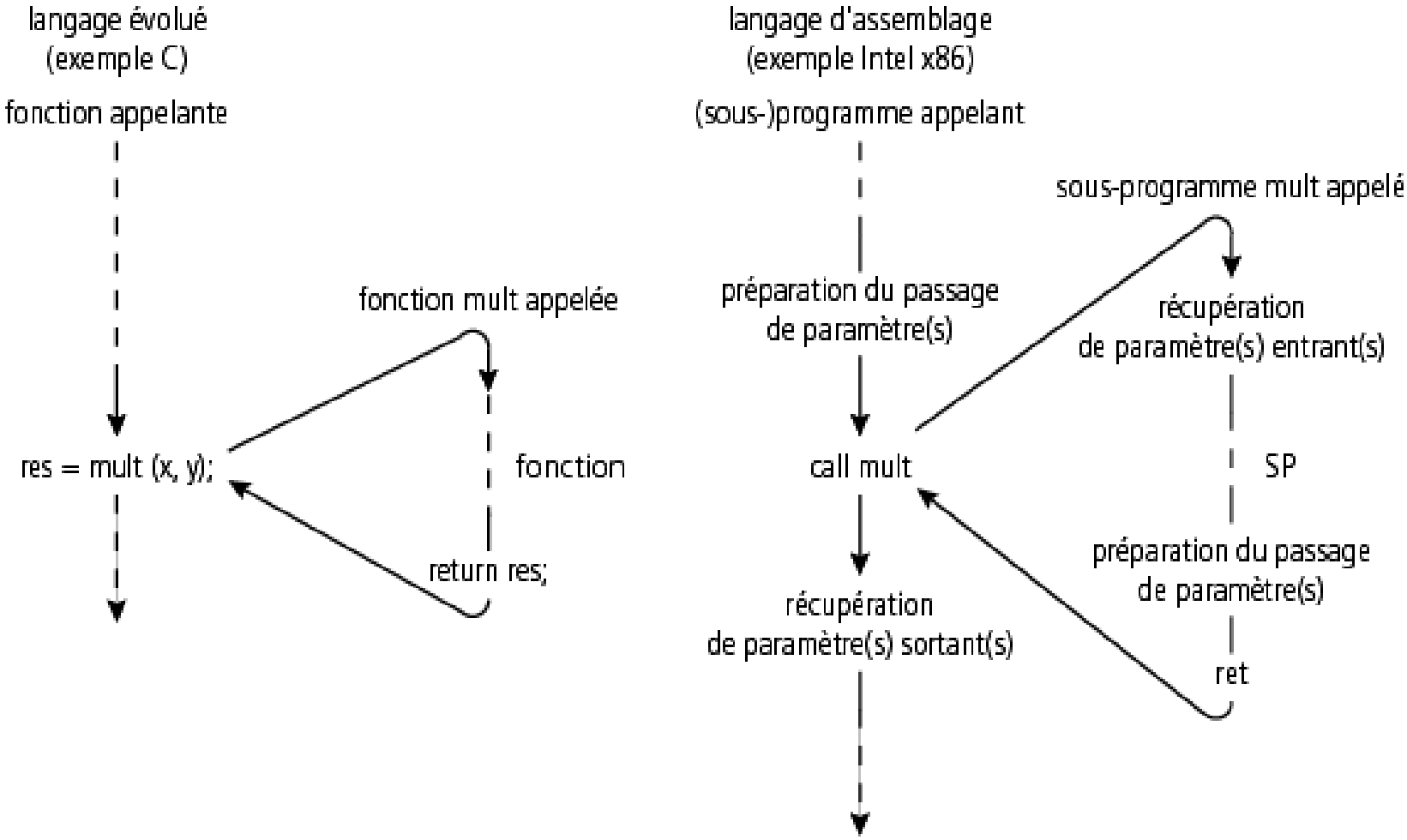
Exemple d'une pile



Un exemple de sous-programme

```
        CODESEG
PROC    mult near
        ; multiplication de deux nombres
        ; passage par valeur et par la pile

        mov bp,sp
        mov bx,[bp+2]
        mov cx,[bp+4]

        mov ax,0

boucle:
        add ax,bx
        dec cx
        jne boucle

; ax contient le r,sultat.
        mov [bp+2],ax

fin:    ret
ENDP    mult

debut:  mov ax,@data
        mov ds,ax
        mov es,ax

;       mov bx,[M]
;       mov cx,[N]

        push [M]
        push [N]

        call mult

;       mov [S],ax
        pop [S]
        pop ax

        mov ah,4ch
        int 21h
        END debut
```

Comparaison avec un langage évolué



L'appel de sous-programme

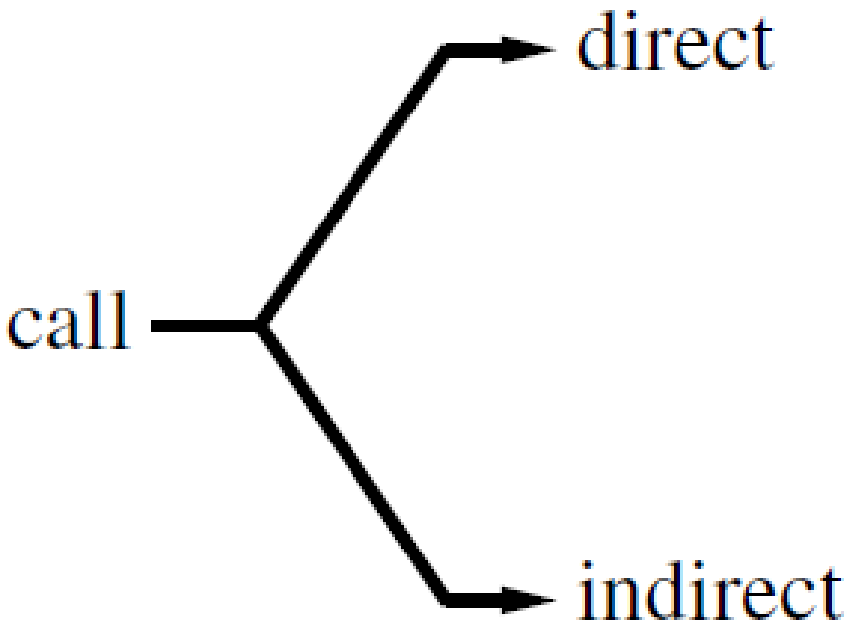
- Syntaxe :

  call <nom_sp>

  ⇒ un opérande : nom du sous-programme

- Opération :
  - appel de sous-programme
- Appels direct et indirect

Appels direct et indirect



intrasegment
- saut relatif proche (± 32 Ko)

intersegment
adressage direct

intrasegment
adressage indirect
- mémoire
- registre

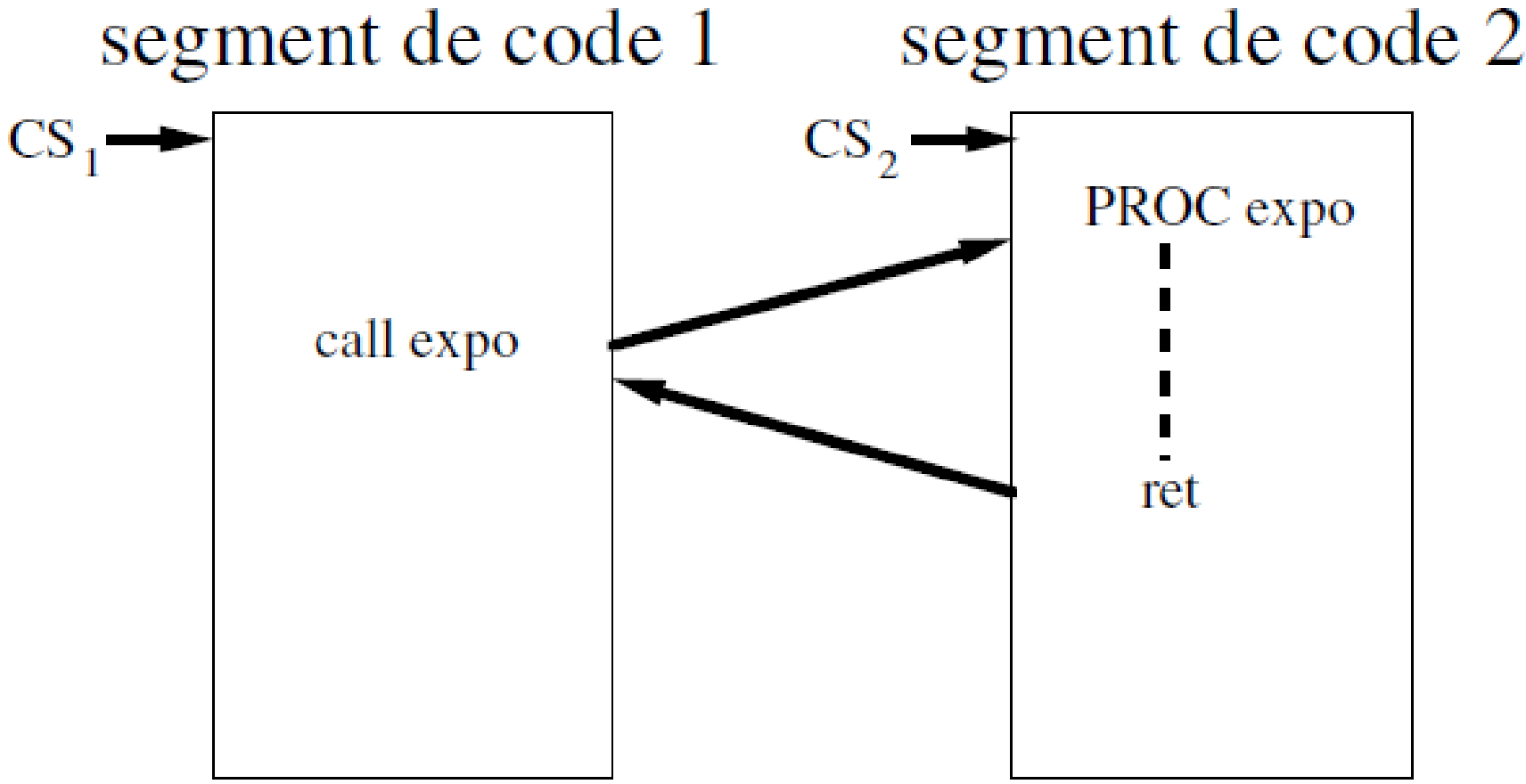
intersegment
adressage indirect mémoire uniquement

Appel direct

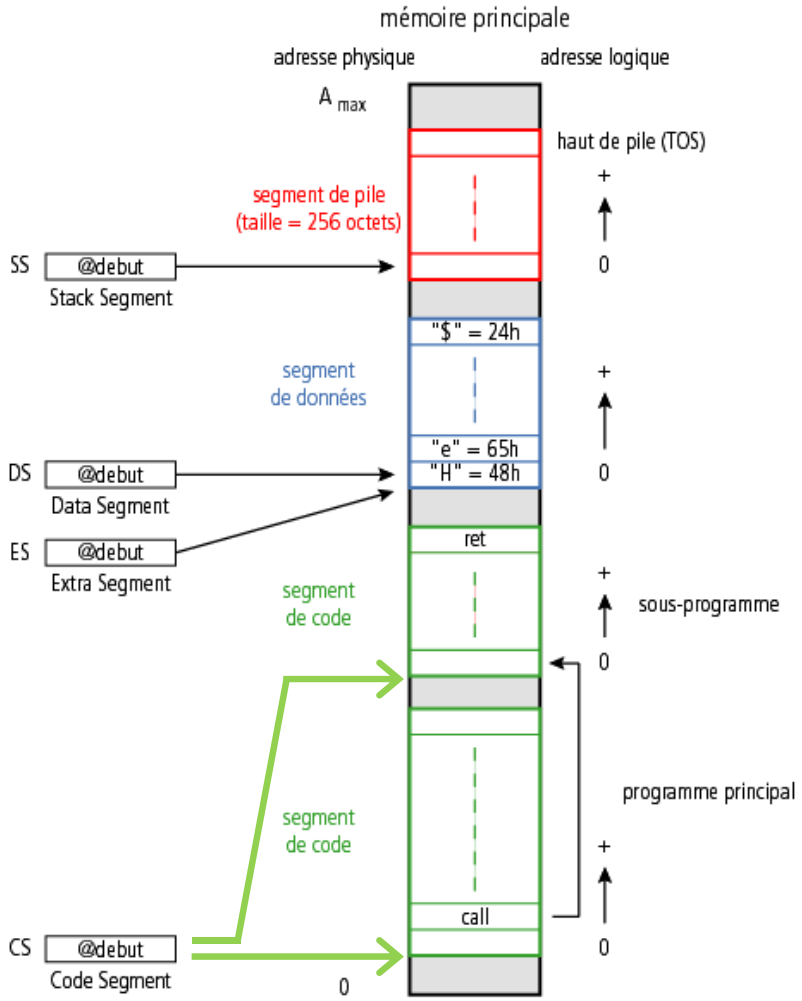
- Syntaxe :

  call <nom_sp>
- Saut relatif par rapport à la position du call
- Format de l'adresse logique : 16 bits
- Longueur maximale du saut : +/- 32 Kio
- Deux types de saut :
  - intra-segment
  - inter-segment

Saut inter-segment



Saut inter-segment



quatre registres de segmentation
du microprocesseur 8086

Pseudo-code

si saut inter-segment

alors

    sp ← (sp) - 2

    [(sp)+1:(sp)] ← (cs)

    cs ← (cs destination)

fin_si

sp ← (sp) - 2

[(sp)+1:(sp)] ← (ip)

ip ← adresse de début de sous-programme

Le retour

- Syntaxe :

  ret
- Aucun opérande
- Opération :
  - retour de sous-programme (*return*) au programme appelant

Pseudo-code

ip ← ([(sp)+1:(sp)])

sp ← (sp) + 2

si saut inter-segment

alors

    cs ← ([(sp)+1:(sp)])

    sp ← (sp) + 2

fin_si

Passage de paramètres, les questions à se poser

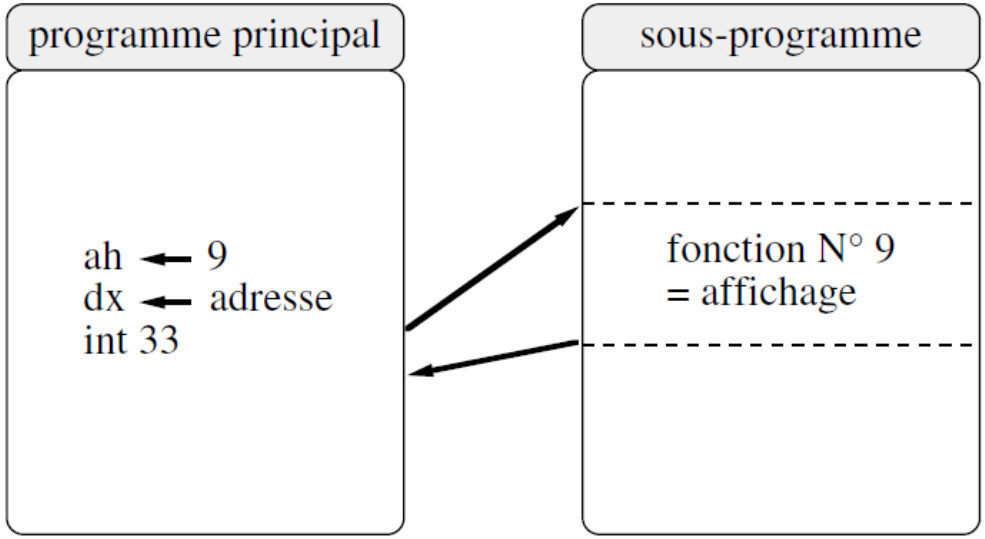
- 1ère question : nombre de paramètres
  - nombre de paramètre(s) entrant(s)
  - nombre de paramètre(s) sortant(s)
- 2ème question : type de passage
  - par valeur
  - par adresse
- 3ème question : format n (bits) des paramètres
  - tenir compte du code utilisé en machine (CBN, $C2^n$, etc,)
  - Prendre en compte l'étendue des valeurs
  - on parle d'un type dans un langage de haut niveau typé qui, lui, prend en compte alors le code

Passage de paramètres, les questions à se poser

- 3ème question : mode de passage (*i.e.* quelle mémoire ?)
  - par registre
  - par la pile
  - par variable mémoire (à déconseiller car effet de bord)

Le problème du passage de paramètres

- Par registre
  - limitation par le nombre
- Par la pile
  - lent
- Par variable mémoire
  - effet de bord
    - non recommandée



Les instructions de manipulation de la pile

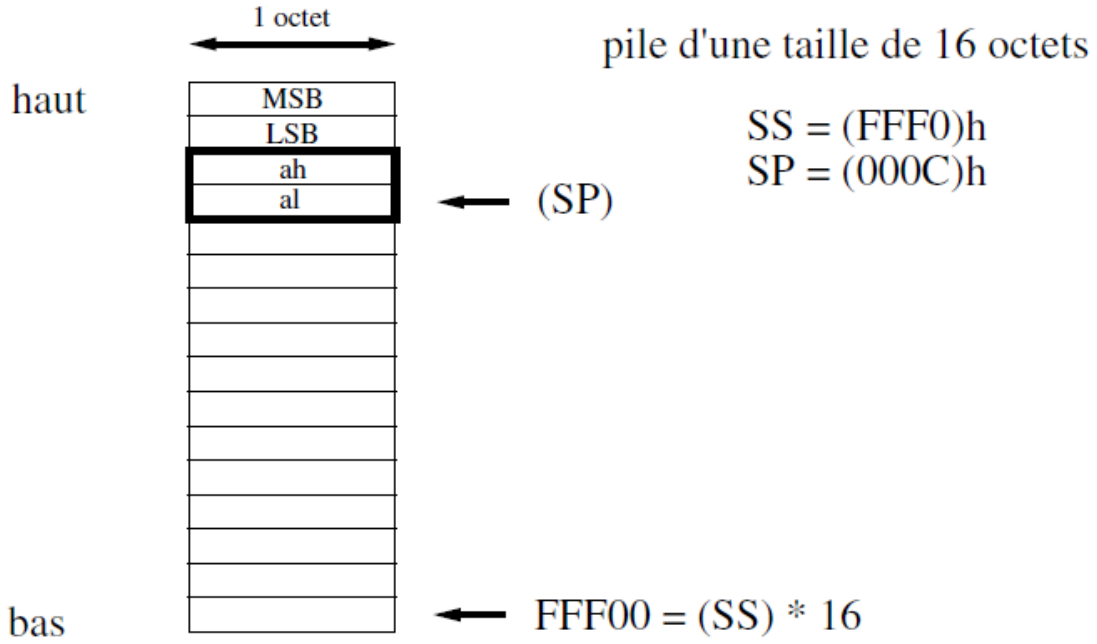
- PUSH : empilement sur la pile
- PUSHF (PUSH Flag)
  - empilement sur la pile du registre des indicateurs
- POP : dépilement de la pile
- POPF (POP Flag)
  - dépilement de la pile du registre des indicateurs

PUSH

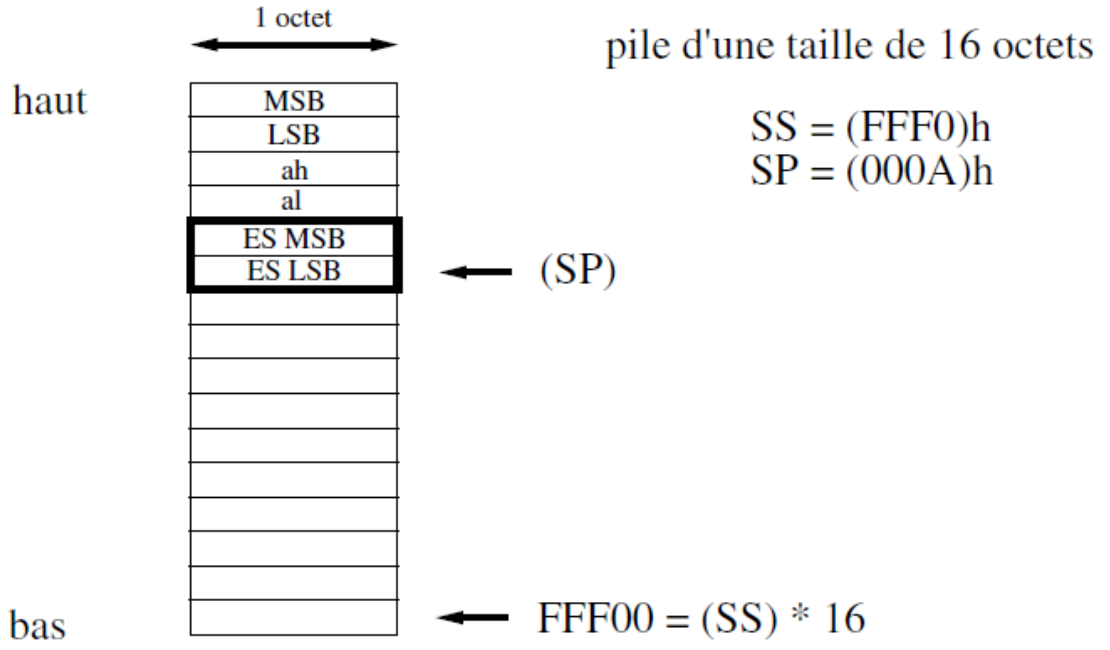
- Un opérande mémoire ou registre
- Opération :
  - empilement du contenu de l'opérande source sur la pile
- Pseudo-code :
  - SP ← (SP) - 2 (en octet)
  - [(SP)] ← (opérande source), mode pointeur = accès indirect

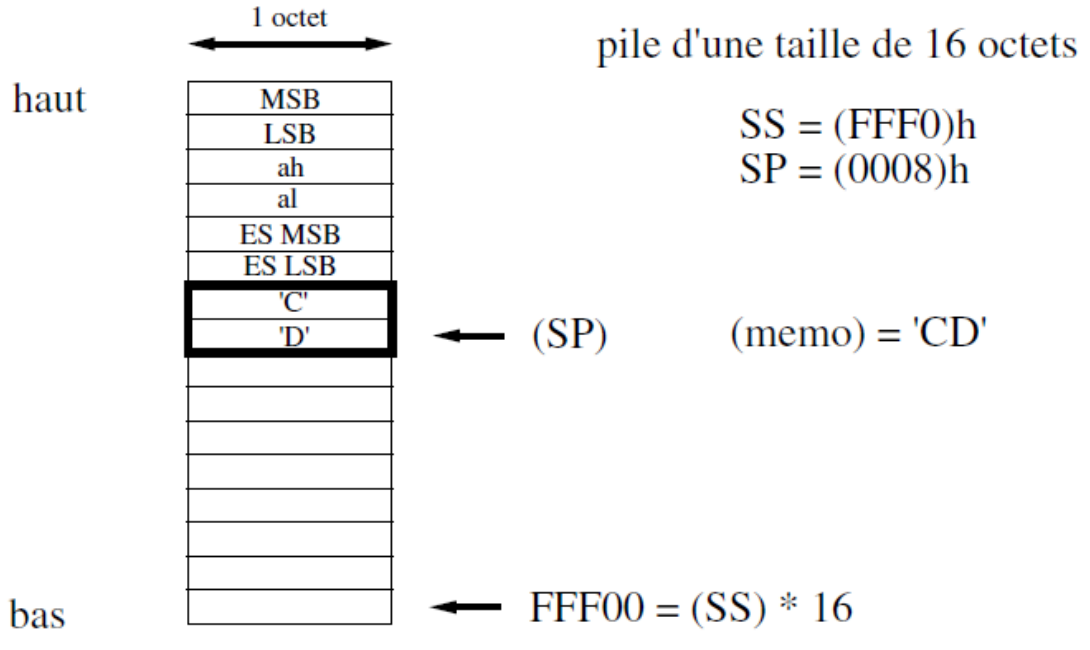
Exemples

- push ax ; empilement de (ax)
- push ES ; empilement de (ES)
- push [memo] ; empilement de (memo)



Exemples (suite)



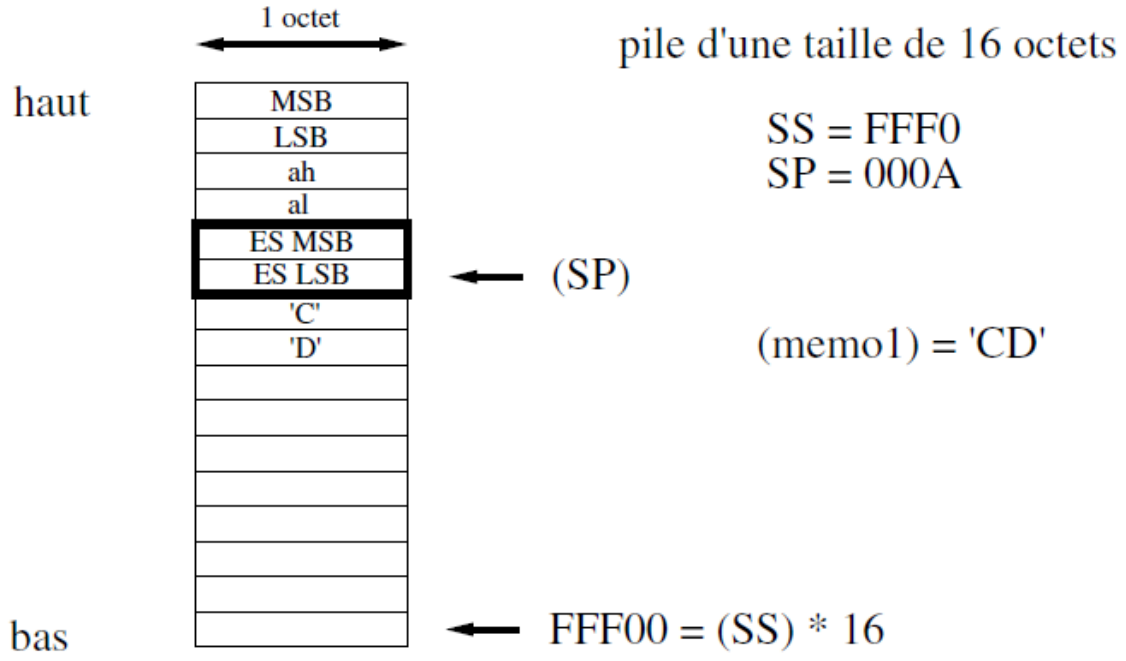


POP

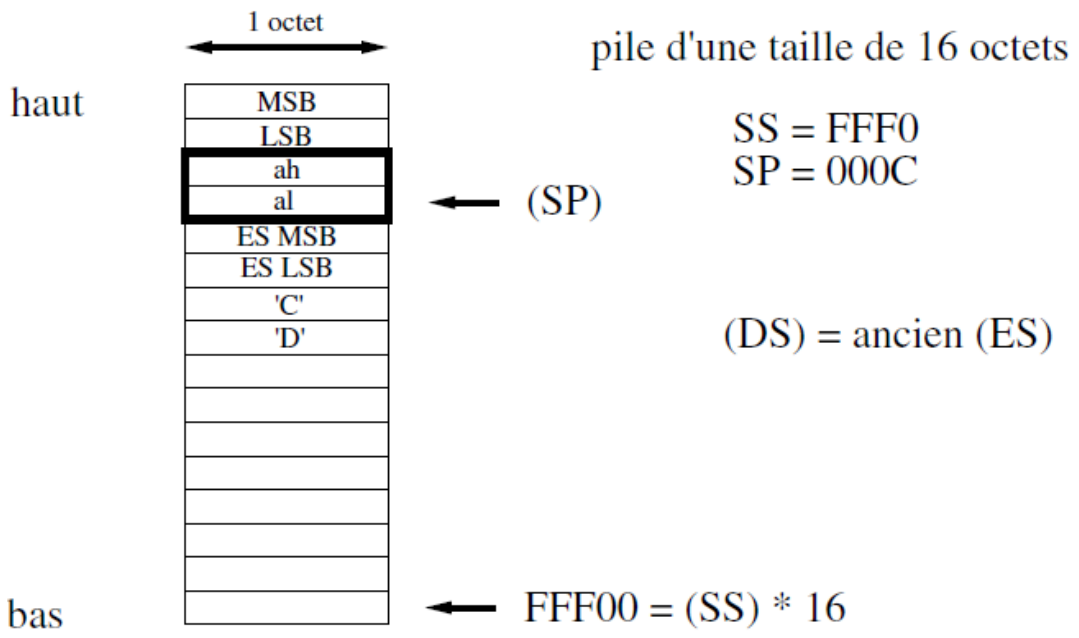
- Un opérande
- Opération :
  - dépilement du contenu de la pile vers l'opérande destination
- Pseudo-code :
  - [opérande destination] ← ([(SP)])
  - SP ← (SP) + 2

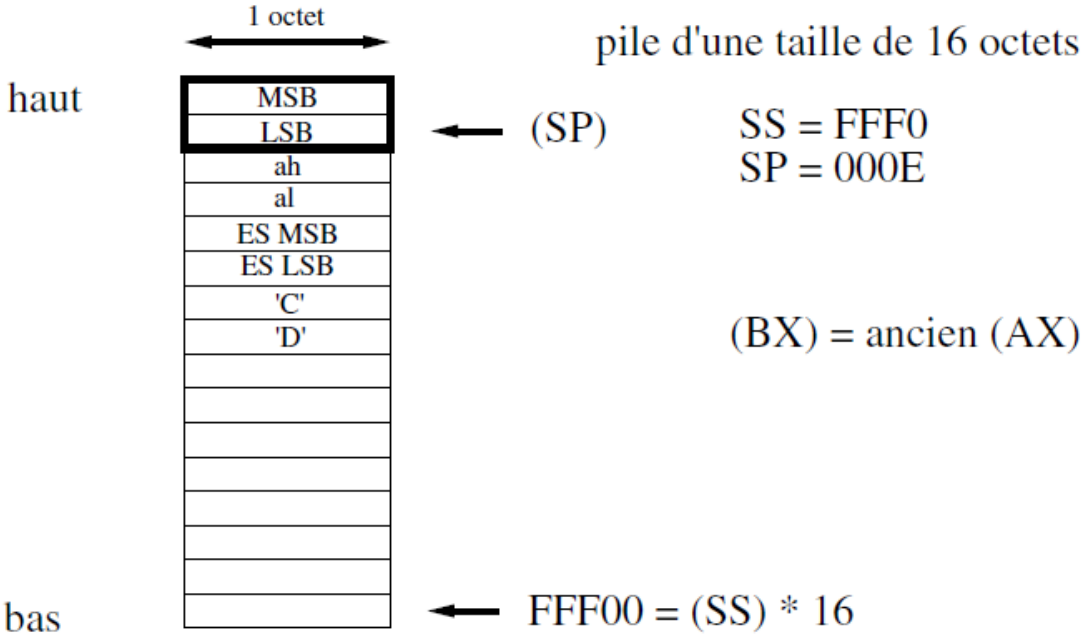
Exemples

- pop [memo1] ; dépilement dans memo1
- pop ds ; dépilement dans ds
- pop bx ; dépilement dans bx



Exemples (suite)





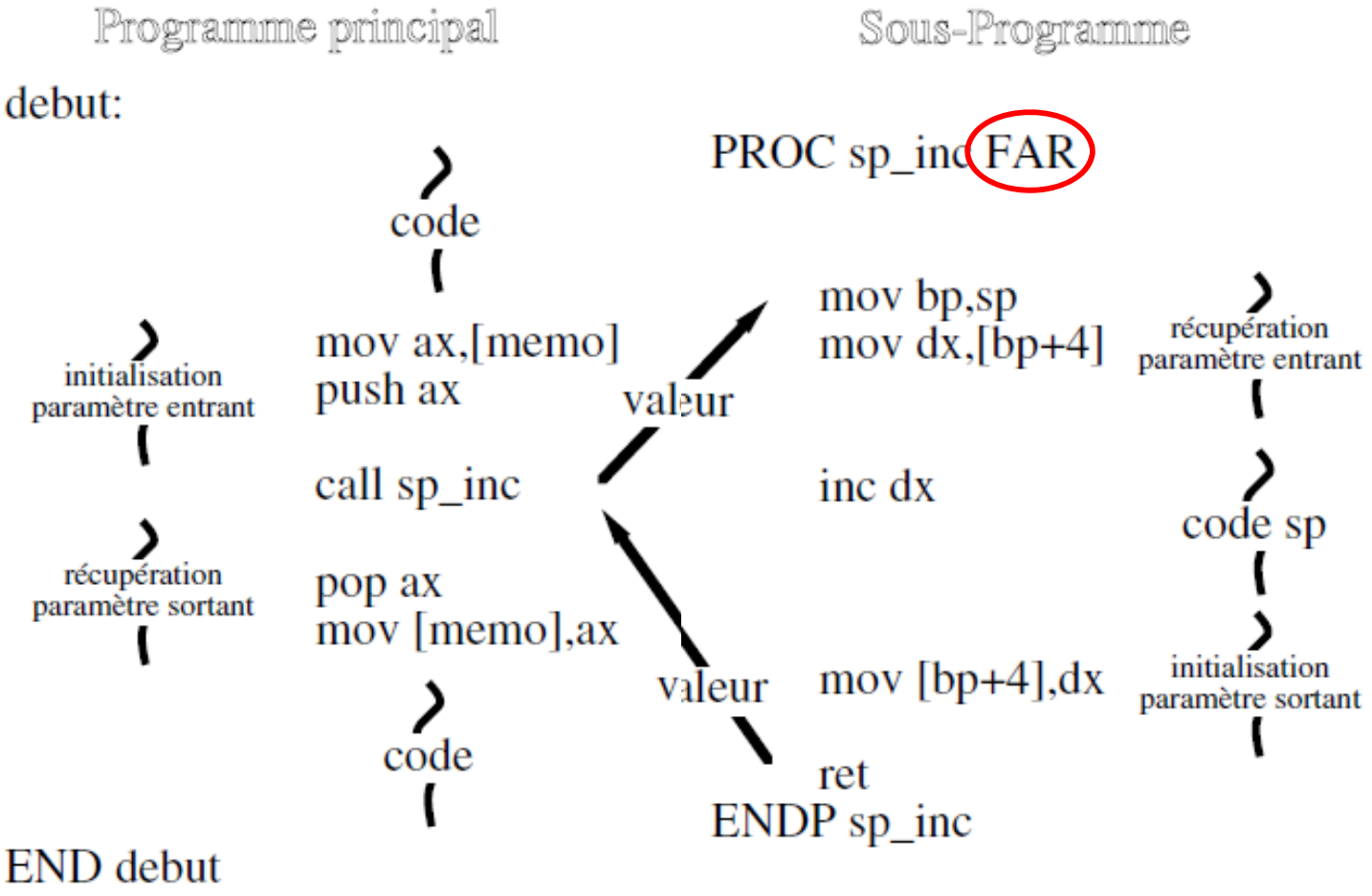
Règle d'or du pointeur de pile du 8086

- Le registre SP pointe la dernière information empilée
  - décrémentation préalable avant tout empilement
  - incrémentation postérieure à tout dépilement
- Autant d'empilements que de dépilements

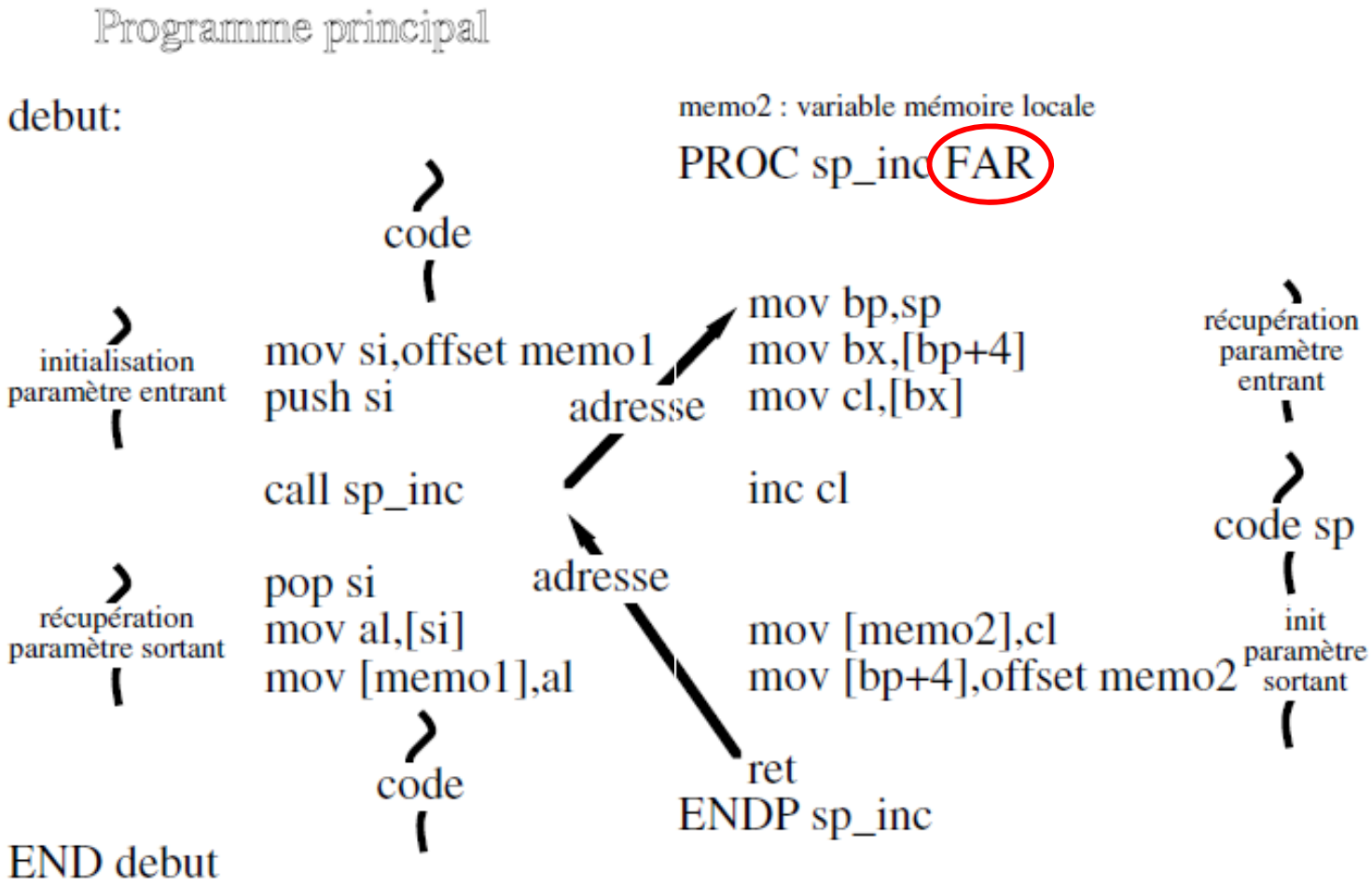
Utilisation de la pile

- Sauvegarde du contexte
  - en premier lieu, l'adresse de retour
- Passage de paramètres
- Déclaration de variables locales
- Accès à des registres non accessibles, directement par des instructions :
  - IP
  - registre des indicateurs

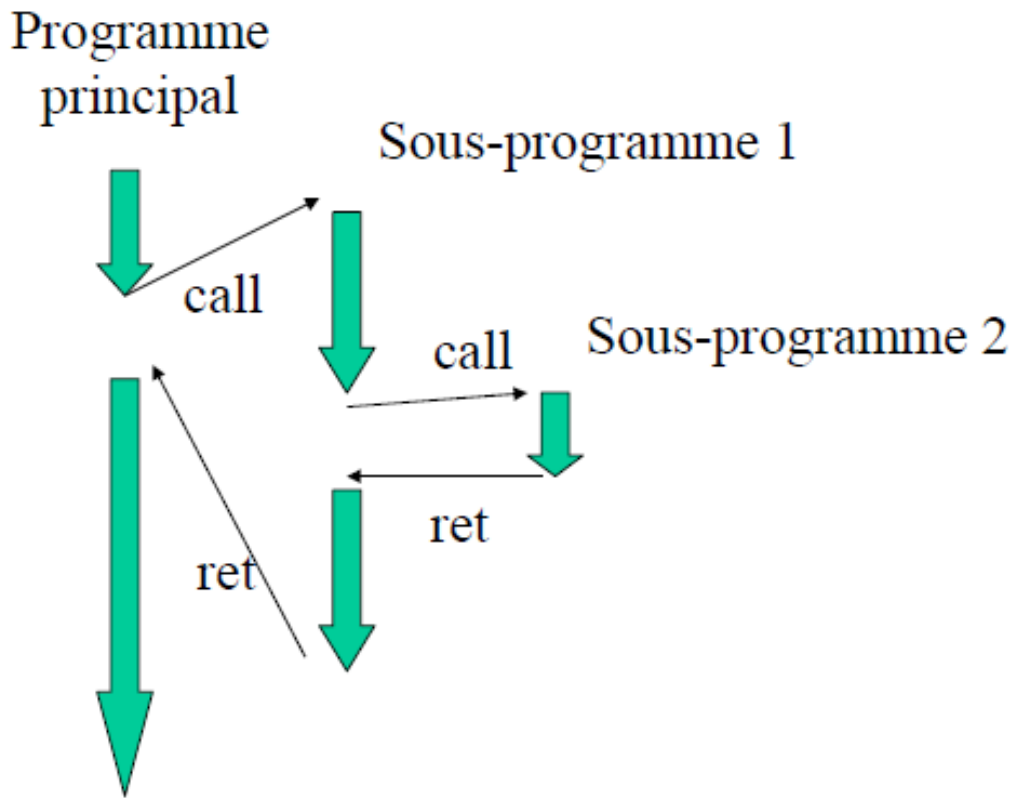
Exemple N° 1 de passage de paramètres par la pile (saut inter-segment)



Exemple N° 2 de passage de paramètres par la pile (saut inter-segment)



Mécanisme cascadable



Conclusion

- Une fonction ou une procédure est un sous-programme en langage d'assemblage
- La notion de pile est fondamentale pour la sauvegarde/restauration du contexte d'exécution et pour le passage des paramètres d'une fonction
- Voir TP correspondant