Pratique du C

Notion de pile en assembleur : définitions théoriques et

Représentation des variables

Appel de fonctio

Passage de

Passage de paramètres par la pile

Passage de paramètre : une copie est faite su la pile

V-2 (04-03-2014)

Pile d'exécution

Notion de pile e assembleur : définitions théoriques et pratiques

des variables automatiques

en C

paramètres par la pile Passage de

paramètre : une copie est faite s la pile

V91 (04-03-2014)

Pile d'exécution

Notion de pile en assembleur : définitions théoriques et pratiques

des variables

Appel de foncti en C

paramètres par

paramètre : une copie est faite si la pile

V91 (04-03-2014)

# Pratique du C Pile d'exécution

Licence Informatique — Université Lille 1
Pour toutes remarques : Alexandre.Sedoglavic@univ-lille1.fr

Semestre 4 — 2015-2016

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

Si x est un élément de E, les relations satisfaites par une pile P et ces opérations sont :

- 1.  $\operatorname{estVide}(P_0) = \operatorname{vrai}$
- 2. supprimer(empiler(x, P)) = P
- 3. estVide(empiler(x, P)) = faux
- **4.** depiler(empiler(x, P)) = x

Cette dernière règle caractérise les piles.

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

# Notion de pile en assembleur : instructions assembleurs associées (Intel 32)

Les modification de la structure de la pile se font par les instructions :

- ▶ push reg : (empiler depuis le registre reg). Lorsque l'on empile un élément sur la pile, l'adresse contenue dans %ESP est décrémentée de 4 octets (car un emplacement de la pile est un mot machine de 32 shannons). En effet, lorsque l'on parcourt la pile de la base vers le sommet, les adresses décroissent.
- ▶ pop reg : (dépiler vers le registre reg). Cette instruction incrémente de 4 octets la valeur de %ESP. Attention, lorsque la pile est vide %ESP pointe sous la pile (l'emplacement mémoire en-dessous de la base de la pile) et un nouveau pop provoquera une erreur.

Il est aussi possible — pour lire ou modifier des valeurs dans la pile — d'utiliser les références mémoire : movl %SS :4(%ESP),%EAX

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

Pratique du C

Notion de pile en essembleur : léfinitions chéoriques et

Représentation des variables automatiques

Passage de paramètres par

Passage de paramètre : une copie est faite sur

assembleur : définitions théoriques et

Notion de pile en assembleur : définitions théoriques et pratiques

Schématiquement, une pile est une structure de données linéaire pour laquelle les insertions et les suppressions d'éléments se font toutes *du même coté*. On parle de structure LIFO : Last In First Out.

Plus formellement, on peut considérer un ensemble d'éléments E et noter  $\mathrm{Pil}(E)$  l'ensemble de toutes les piles sur E. Par exemple, les entiers peuvent constituer l'ensemble E; la pile vide  $P_0$  est dans  $\mathrm{Pil}(E)$ . Les opérations usuelles sur une pile sont :

- estVide est une application de Pil(E) dans (vrai, faux), estVide(P) est vrai si, et seulement si, P est la pile  $P_0$ .
- ▶ empiler est une application de  $E \times Pil(E)$  dans Pil(E).
- ▶ depiler est une application de  $Pil(E) \setminus P_0$  dans E.
- Supprimer est une application de Pil(E) \ P<sub>0</sub> dans Pil(E).

# Implantation d'une pile (architecture Intel 32)

Un segment de la mémoire est dévolu à la pile. Les registres %SS et %ESP sont deux registres servant à gérer la pile :

**%SS** (Stack Segment i.e. segment de pile) est un registre 16 bits contenant l'adresse du segment de pile courant;

L'assembleur vous fera manipuler une pile qui est stockée "en fond de panier", c.à.d dans les adresses les plus hautes de la mémoire. Ainsi, la base de la pile se trouve à l'adresse maximale, et elle s'accroit vers les adresses basses.

**%ESP** (Stack Pointer i.e. pointeur de pile) est le déplacement pour atteindre le sommet de la pile.

Ainsi, %ESP pointe sur le dernier mot machine occupé de la pile en mémoire.

PUSH POP

RX3 est %ESP;

%ESP pointe sur l'octet venant d'être empilé;

On empile un mot machine (4 octets).

FFFF

Pratique du C

Notion de pile en assembleur : définitions théoriques et

Représentation des variables

Appel de fonction en C

Passage de paramètres par la

Passage de paramètre : une copie est faite su la pile

V91 (04-03-2014)

Pratique du C Pile d'exécution

Notion de pile en assembleur : définitions théoriques et pratiques

des variables automatiques

en C
Passage de

Passage de paramètre : une copie est faite su

V91 (04-03-2014)

Pile d'exécution

Notion de pile en

définitions théoriques et pratiques Représentation des variables

en C

Passage de paramètres par la pile

paramètre : une copie est faite su la pile

V91 (04-03-2014)

Les variables *automatiques* (locales à une fonction) sont stockées dans la pile.

Le registre %EBP (Base Pointer) contient un déplacement correspondant à une position dans la pile.

Il sert à pointer sur une donnée dans la pile.

Représentation d'une variable automatique dans la pile :

```
int globale = 7;
                           .globl globale .data
                  globale: .long
                           .text
                            .globl main .type main,@function
main
                     main:
(void)
                            movl
                                     %esp, %ebp
  int locale;
                            subl
                                     $8, %esp
  locale = 1 ;
                            . . . .
 return 0 ;
                                     $1, -4(%ebp)
                            movl
```

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

# Représentation de variables de types différents et manipulation

Peu importe le type des variables automatiques que l'on veut représenter, la méthode est la même :

```
text
                               .globl main
main
(void)
                               .type main, @function
                         main: ....
  struct Gauss{
                                movl %esp, %ebp
   int re ;
                                subl $24, %esp
    int im ;
                                . . . .
  } var = {
                                movl
                                        $1, -8(%ebp)
   .re = 1 ,
                                       $1, -4(%ebp)
                                movl
    .im = 1 ,
                                       $97, -10(%ebp)
                                movb
                                movb
                                        $98, -9(%ebp)
                                        -9(%ebp), %dl
                                movb
  char tab[2] = {'a', 'b'};
                                leal
                                        -10(%ebp), %eax
 tab[0] += tab[1] ;
                                addb
                                       %dl. (%eax)
  return 0 ;
```

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

Pratique du C Pile d'exécution

Appel de fonction en C

### Les instructions assembleur call et ret

L'appel d'une *routine* se fait par call label\_routine. Soit un code implanté à l'adresse 1000 et une routine à l'adresse 1100.

```
1000 mov $1,%eax +---> label: 1100 shl $1,%eax
1002 mov $3,%ebx | 1102 add %ebx,%eax
1004 CALL label ------| 1104 and $7,%eax
1007 mov $2,%eax <-----| 1106 add '0',%eax
1009 int 0x80 | 1108 RET
```

Le sous-programme doit contenir l'instruction RET qui permet de revenir au programme appelant.

Lors du CALL, %EIP reçoit la valeur 1100, adresse de la prochaine instruction à exécuter, tandis que l'adresse de retour 1007 est empilée sur la pile.

Sur le  $_{\rm RET}$ , le sommet de pile de valeur 1007 est dépilé, et son contenu est rangé dans  $\%{\rm EIP}.$ 

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

#### Pratique du C Pile d'exécution

lotion de pile en ssembleur : léfinitions héoriques et

Représentation des variables automatiques

Passage de paramètres par la

pile
Passage de

Passage de paramètre : une copie est faite sur la nile

## Représentation des variables automatiques

movl %esp, %ebp

%ESP data 4 octets

FFFF bas de pile

%EBP mis en place

subl	\$8, %esp
%ESP	vide
	vide
%EBP	data
	4 octets
	:
FFFF	bas de pile

réserver de l'espace

 mov1
 \$1, -4(%ebp)

 %ESP
 vide

 1
 %EBP

 4 octets
 :

 FFFF
 bas de pile

affecter la variable variable

# Rappel sur les registres associés à l'exécution du code

Le code exécutable d'un programme se présente sous la forme d'une suite finie contigüe d'octets stockée dans un segment de la mémoire.

Le registre %CS (Code Segment). Ce registre 16 bits contient le numéro du segment mémoire dans lequel sont stocké les instructions assembleur du code à exécuter. On ne peut pas accéder directement à ce registre.

Le registre %EIP (Instruction Pointer). Le registre %EIP contient l'offset de la prochaine instruction à exécuter. Il est modifié automatique à chaque exécution et peut être manipulé par des instruction du type jmp, call, ret, etc. On ne peut pas accéder directement à ce registre.

+□ > ←□ > ← = > = − 9

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf

### Appel de fonction en C (sans paramètre)

```
.text
int.
                                     .globl UN
IJМ
                                UN:
(void)
                                              $1, %eax
                                     movl
  return 1 :
                                     ret
                                    .globl main
                              main:
int
main
(void)
                                     movl
                                              %esp, %ebp
                                              $8, %esp
                                     subl
  int var ;
                                     . . . .
  var = UN() ;
                                     call
                                              %eax, -4(%ebp)
  return 0 :
                                     movl
1
                                     movl
                                              $0, %eax
/* Remarquez que la valeur
   de retour transite par
                                     ret
   le registre %eax
```

# Appel de fonction en C

V91 (04-03-2014)

V91 (04-03-2014)

paramètres par la pile

V91 (04-03-2014)

## Ce qui se passe sur la pile

%ESP	vide	
	var. aut.	
%EBP	data	
	4 octets	
FFFF	bas de pile	
avant le call		

	%ESP	adresse		
		de retour		
		vide		
		var. aut.		
	%EBP	data		
		4 octets		
<b>+</b>	FFFF	bas de pile		

pendant le call

$\downarrow$
vide
var. aut.
data
4 octets
bas de pile

après le ret

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

```
int
                           .text
PlusUn
                           .globl PlusUn
(int par)
                   PlusUn:
                           pushl
                                    %ebp
  return par+1;
                           movl
                                    %esp, %ebp
                                    8(%ebp), %eax
                           movl
                           incl
                                    %eax
                           leave ret
                            .globl main
                     main:
                           pushl
                                    %ebp
                                    %esp, %ebp
int
                           movl
main
                            subl
                                    $8, %esp
(void)
                           movl
                                    $7, -4(%ebp)
                                    $12, %esp
                           subl
                           pushl
                                    -4(%ebp)
  int var ;
  var = 7;
```

PlusUn

\$16, %esp

call

addl

leave ret

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

# Ce qui se passe sur la pile

return PlusUn(var) ;

%EBP_old
adresse
de retour
4 octets
bas de pile

	%ESP	
		var. loc.
	%EBP1	%EBP_old
		adresse
		de retour
ľ		4 octets
	FFFF	bas de pile

création de variables placement du nouveau pointeur de base automatiques

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

paramètres par la pile

Comme toutes variables automatiques, les paramètres sont stockés dans la pile. Dans une fonction C les variables et les paramètres sont gérés en suivant les étapes :

- 1. Sauver le pointeur de base de pile courant sur la pile;
- 2. Se donner un nouveau pointeur de base de pile;
- 3. Déclarer les variables automatiques sur la pile; En cas d'appel de fonction avec passage de paramètres :
- 4. Empiler les paramètres sur la pile;
- 5. Effectuer un call (qui empile automatiquement l'adresse de retour sur la pile);
  - Dans la fonction appelée, on peut utiliser l'espace de pile associée aux paramètres; Cette fonction se termine par un ret (qui dépile automatiquement l'adresse de
- 6. Suprimer l'espace de pile maintenant inutile associé aux paramètres.

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf

# Ce qui se passe sur la pile

%ESP	adresse		
	de retour		
	4 octets		
FFFF	bas de pile		

%ESP	%EBP_old	
	adresse	
	de retour	
	4 octets	
FFFF	bas de pile	

début de fonction

empilement de l'ancien pointeur de base

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf

%ESP paramètres var. loc. %EBP1 %EBP\_old adresse de retour 4 octets **FFFF** bas de pile

empilement	de	paramètres
CHIPHCHICH	uc	parametro

%ESP	adresse	
	de retour	
	paramètres	
	var. loc.	
%EBP1	%EBP_old	
	adresse	
	de retour	
	4 octets	
FFFF	bas de pile	

FFF	ba	as	de	p
après	un	С	all	

#### Pratique du C Pile d'exécution

Notion de pile en assembleur : définitions théoriques et

Représentation des variables

Appel de fonction

Passage de paramètres par la pile

Passage de paramètre : une copie est faite sui

V91 (04-03-2014)

Pratique du C Pile d'exécution

Notion de pile de assembleur : définitions théoriques et

Représentation des variables automatiques

en C

paramètres par l pile

Passage de paramètre : une copie est faite sur la pile

V91 (04-03-2014)

## Pile d'exécution

Notion de pile er assembleur : définitions théoriques et pratiques

automatiques

Passage de

pile

Passage de paramètre : une copie est faite sur la pile

V91 (04-03-2014)

Pile (fonction appellée)

%ESP	var. loc.
%EBP	%EBP1
	adresse
	de retour
	paramètres
	var. loc.
%EBP	%EBP_old
	adresse
	de retour
	:

2) l'instruction ret dépile l'adresse de retour et positionne le registre pointeur d'instruction à cette adresse.

À la fin de la fonction appellée 1) une instruction leave permet d'enlever de la pile l'espace associé aux variables automatiques et au stockage du pointeur de base. De plus, elle réaffecte au registre %EBP la valeur du pointeur de base de la fonction appelante.

3) il ne reste plus qu'à supprimer de la pile l'espace associé aux paramètres (addl \$16,%esp) pour se retrouver dans la situation d'avant l'appel de fonction.

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

### Pratique du Pile d'exécuti

```
.globl main
int
main
                           main:
(void)
                                 pushl %ebp
                                 movl %esp, %ebp
                                 subl $8, %esp
   int a = 1:
  int b = 2 ;
                                 andl $-16, %esp
  PER(a,b);
                                 movl $1, -4(%ebp)
   return 0 ;
                                 movl $2, -8(%ebp)
                                 subl $8, %esp
                                 pushl -8(%ebp)
                                 pushl -4(%ebp)
/* certains compilateurs
                                                                   Passage de
paramètre : une
copie est faite sur
la pile
placent variables
                                 call PER
automatiques et param\'etres
                                 addl $16, %esp
aux m\^emes endroits
                                 movl $0, %eax
(pas de push) */
                                 leave
                                 ret.
```

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf V91 (04-03-2014)

# La fonction appelante

```
.globl main
                   main:
                          pushl %ebp
                          movl %esp, %ebp
                          subl $8, %esp
                          andl $-16, %esp
int
                          movl $1, -4(%ebp)
main
                          movl $2, -8(%ebp)
                          subl $8, %esp
(void)
                          leal -8(%ebp), %eax
                          pushl %eax
  int a ;
  int b;
                          leal -4(%ebp), %eax
   a = 1 ;
                         pushl %eax
  b = 2;
                          call PER
                          addl $16, %esp
  PER(&a.&b) :
                          movl $0. %eax
   return 0;
                          leave
                          ret
```

www.fil.univ-lille1.fr/~sedoglav/C/Cours09.pdf

## Pratique du C

Notion de pile en assembleur : définitions théoriques et pratiques

les variables automatiques

Passage de paramètres par la

Passage de paramètre : une copie est faite sur la pile

## Exemple incorrect de permutation

```
.text
                                    .globl PER
void
                            PER:
PER
                                    pushl %ebp
(int alpha, int beta)
                                    movl %esp, %ebp
                                    subl $4, %esp
                                    movl 8(%ebp), %eax
  int tmp ;
  tmp = alpha ;
                                    movl %eax, -4(%ebp)
  alpha = beta ;
                                    movl
                                          12(%ebp), %eax
  beta = tmp;
                                    movl %eax, 8(%ebp)
  return ;
                                    movl
                                          -4(%ebp), %eax
                                    movl
                                          %eax, 12(%ebp)
                                    leave
```

# Passage de paramètre par adresse : les adresses sont copiées sur la pile

```
.globl PER
                          PER:
void
                                 pushl %ebp
PER
                                 movl %esp, %ebp
(int *alpha, int *beta)
                                 subl $24, %esp
                                 movl 8(%ebp), %eax
 int tmp;
                                 movl (%eax), %eax
  tmp = *alpha ;
                                 movl
                                       %eax, -12(%ebp)
  *alpha = *beta ;
                                 movl 12(%ebp), %eax
  *beta = tmp ;
                                 movl
                                       (%eax), %edx
                                movl 8(%ebp), %eax
 return :
                                 movl %edx, (%eax)
                                 movl 12(%ebp), %edx
                                 movl -12(%ebp), %eax
                                 movl %eax, (%edx)
                                 leave
                                 ret.
```