Rendu 2 : Compilation

Maxime BRASLEY / Alexis LOPES VAZ / Gabin HEMMERLE / Hawa DIABATE

Partie 1:

Code Assembleur Attendu	Code Assembleur Obtenu

Arbre 1:
CMOVE(pile,SP)
BR(debut)
 	
main:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (0)
DEALLOCATE (0)
POP (BP)
POP (LP)
RTN()
 	
debut:
CALL(main)
HALT()
pile:
	`| Code genere par le compilateur`
	`.include beta.uasm`
	`.include intio.uasm`
	`.options tty`
	
	`CMOVE(pile,SP)`
	`BR(debut)`
	
	`main:`
	`PUSH (LP)`
	`PUSH (BP)`
	`MOVE (SP, BP)`
	`ALLOCATE (0)`
	`DEALLOCATE (0)`
	`POP (BP)`
	`POP (LP)`
	`RTN()`
	
	`debut:`
	`CALL(main)`
	`HALT()`
	`pile:`
	`Test #1`
	
	`nom : main; type : void; categorie : fonction;`

Arbre 2:
CMOVE(pile,SP)
BR(debut)
i: LONG(10)
j: LONG(20)
k: LONG(0)
l: LONG(0)
 	
main:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (0)
DEALLOCATE (0)
POP (BP)
POP (LP)
RTN()
 	
debut:
CALL(main)
HALT()
pile:
	`| Code genere par le compilateur`
	`.include beta.uasm`
	`.include intio.uasm`
	`.options tty`
	
	`CMOVE(pile,SP)`
	`BR(debut)`
	`i: LONG(10)`
	`j: LONG(20)`
	`k: LONG(0)`
	`l: LONG(0)`
	
	`main:`
	`PUSH (LP)`
	`PUSH (BP)`
	`MOVE (SP, BP)`
	`ALLOCATE (0)`
	`DEALLOCATE (0)`

```
POP (BP)
POP (LP)
RTN()

debut:
CALL(main)
HALT()
pile:
  Test #2

  nom : main; type : void; categorie : fonction;

  nom : i; type : int; categorie : global; valeur : 10;

  nom : j; type : int; categorie : global; valeur : 20;

  nom : k; type : int; categorie : global;

  nom : l; type : int; categorie : global;
```

Arbre 3 :

```
            CMOVE(pile,SP)
            BR(debut)
i: LONG(10)
j: LONG(20)
k: LONG(0)
l: LONG(0)

main:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (0)

|k
CMOVE(2, r0)
PUSH (r0)
POP (r0)
ST(r0, k)

|i
LD(i, r0)
PUSH (r0)

|j
LD(j, r0)
PUSH (r0)

|3
CMOVE(3, r0)
PUSH (r0)

mul 3 j
POP (r2)
POP (r1)
MUL (r1, r2, r3)
PUSH (r3)

|addition
POP (r2)
POP (r1)
ADD (r1, r2, r3)
PUSH (r3)

|l
POP (r0)
ST(r0, l)
DEALLOCATE (0)
POP (BP)
POP (LP)
RTN()
```

```
| Code genere par le compilateur
.include beta.uasm
.include intio.uasm
.options tty

   CMOVE(pile,SP)
   BR(debut)
i: LONG(10)
j: LONG(20)
k: LONG(0)
l: LONG(0)

main:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (0)
CMOVE(2, r0)
PUSH (r0)
POP (r0)
ST(r0, k)
LD(i, r0)
PUSH (r0)
LD(j, r0)
PUSH (r0)
CMOVE(3, r0)
PUSH (r0)
POP (r2)
POP (r1)
MUL (r1, r2, r3)
PUSH (r3)
POP (r2)
POP (r1)
ADD (r1, r2, r3)
PUSH (r3)
POP (r0)
ST(r0, l)
DEALLOCATE (0)
POP (BP)
POP (LP)
RTN()
```

debut: CALL(main) HALT() pile:	debut: CALL(main) HALT() pile: nom : main; type : void; categorie : fonction; nom : i; type : int; categorie : global; valeur : 10; nom : j; type : int; categorie : global; valeur : 20; nom : k; type : int; categorie : global; nom : l; type : int; categorie : global;

Arbre 4:
CMOVE(pile,SP)
BR(debut)
i: LONG(0)
j: LONG(20)
 	
main:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (0)
 	
|lis i
RDINT()
PUSH (r0)
POP (r0)
ST(r0, i)
 	
|i + j
LD(i, r0)
PUSH (r0)
LD(j, r0)
PUSH (r0)
POP (r2)
POP (r1)
ADD (r1, r2, r3)
PUSH (r3)
 	
|ecrire
POP (r0)
WRINT()
DEALLOCATE (0)
POP (BP)
POP (LP)
RTN()
 	
debut:
CALL(main)
HALT()
pile:
	| Code genere par le compilateur
	.include beta.uasm
	.include intio.uasm
	.options tty
	
	CMOVE(pile,SP)
	BR(debut)
	i: LONG(0)
	j: LONG(20)
	
	main:
	PUSH (LP)
	PUSH (BP)
	MOVE (SP, BP)
	ALLOCATE (0)
	RDINT()
	PUSH (r0)
	POP (r0)
	ST(r0, i)
	LD(i, r0)
	PUSH (r0)
	LD(j, r0)
	PUSH (r0)
	POP (r2)
	POP (r1)
	ADD (r1, r2, r3)
	PUSH (r3)
	POP (r0)
	WRINT()
	DEALLOCATE (0)
	POP (BP)
	POP (LP)
	RTN()
	
	debut:
	CALL(main)
	HALT()
	pile:
	nom : i; type : int; categorie : global;
	nom : j; type : int; categorie : global; valeur : 20;
	nom : main; type : void; categorie : fonction;

Arbre 5: CMOVE(pile,SP) BR(debut) i: LONG(0)	| Code genere par le compilateur .include beta.uasm .include intio.uasm

```
main:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (0)

|i
RDINT()
PUSH (r0)
POP (r0)
ST(r0, i)

si:
|comparaison
LD(i, r0)
PUSH (r0)
CMOVE(10, r0)
PUSH (r0)
POP (r2)
POP (r1)
CMPLT (r1, r2, r3)
PUSH (r3)
POP (r0)
BF(R0, sinon)

alors:
|ecris 1
CMOVE(1, r0)
PUSH (r0)
POP (r0)
WRINT()
BR(fsi)

sinon:
|ecris 2
CMOVE(2, r0)
PUSH (r0)
POP (r0)
WRINT()

fsi:
DEALLOCATE (0)
POP (BP)
POP (LP)
RTN()

debut:
CALL(main)
HALT()
pile:
```

```
.options tty

   CMOVE(pile,SP)
   BR(debut)
i: LONG(0)

main:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (0)
RDINT()
PUSH (r0)
POP (r0)
ST(r0, i)
si_1:
LD(i, r0)
PUSH (r0)
CMOVE(10, r0)
PUSH (r0)
POP (r2)
POP (r1)
CMPLT (r1, r2, r3)
PUSH (r3)
POP (r0)
BF(R0, sinon_1)
alors_1:
CMOVE(1, r0)
PUSH (r0)
POP (r0)
WRINT()
BR(fsi_1)
sinon_1:
CMOVE(2, r0)
PUSH (r0)
POP (r0)
WRINT()
fsi_1:
DEALLOCATE (0)
POP (BP)
POP (LP)
RTN()

debut:
CALL(main)
HALT()
pile:
 nom : i; type : int; categorie : global;
 nom : main; type : void; categorie : fonction;
```

Arbre 6:

```
CMOVE(pile,SP)
BR(debut)

|data

 i : LONG(0)
n : LONG(5)
```

```
| Code genere par le compilateur
.include beta.uasm
.include intio.uasm
.options tty

   CMOVE(pile,SP)
   BR(debut)
i: LONG(0)
```

```
|instructions

main:
        PUSH(LP)
        PUSH(BP)
        MOVE(SP,BP)
        ALLOCATE(0)
        CMOVE(0, R0)
        PUSH(R0)
        POP(R0)
        ST(R0, i)
                                BR(tq)
    tq:
        LD(i , R0)
        PUSH(R0)
        LD(n,R0)
        PUSH(R0)
        POP(R1)
        POP(R2)
        CMPLT(R2,R1,R0)
        PUSH(R0)
        POP(R0)
        BF(R0, ftq_1)
                                BR(alors)
alors :
        PUSH(R0)
        POP(R0)
        WRINT()
        LD(i, R0)
        PUSH(R0)
        CMOVE(1,  R0)
        PUSH(R0)
        POP(R2)
        POP(R1)
        ADD(R2,R1,R0)
        PUSH(R0)
        POP(R0)
        ST(RO, i )
                                BR(tq)
 ftq_1:
        BR(ret_main)
ret main :
         DEALLOCATE(0)
         POP(BP)
         POP(LP)
         RTN()
debut :
         CALL(main)
         HALT()
          pile:
```

```
h: LONG(5)

main:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (0)
CMOVE(0, r0)
PUSH (r0)
POP (r0)
ST(r0, i)
tantque_1:
LD(i, r0)
PUSH (r0)
LD(h, r0)
PUSH (r0)
POP (r2)
POP (r1)
CMPLT (r2, r1, r3)
PUSH (r3)
POP (R0)
BF(R0, ftantque_1)
LD(i, r0)
PUSH (r0)
POP (r0)
WRINT()
LD(i, r0)
PUSH (r0)
CMOVE(1, r0)
PUSH (r0)
POP (r2)
POP (r1)
ADD (r1, r2, r3)
PUSH (r3)
POP (r0)
ST(r0, i)
BR(tantque_1)
ftantque_1:
DEALLOCATE (0)
POP (BP)
POP (LP)
RTN()

debut:
CALL(main)
HALT()
pile:
 nom : i; type : int; categorie : global;
 nom : h; type : int; categorie : global; valeur : 5;
 nom : main; type : void; categorie : fonction;
```

Arbre 7:

```
CMOVE(PILE,SP)
BR(debut)

|data

Main:

PUSH(LP)
PUSH(BP)
```

```
| Code genere par le compilateur
.include beta.uasm
.include intio.uasm
.options tty

   CMOVE(pile,SP)
   BR(debut)
a: LONG(10)

f:
```

```
MOVE(SP,BP)
ALLOCATE(0)

CMOVE(3,R0)
PUSH(R0)
CALL(f)
DEALLOCATE(1)

LD(a,R0)
PUSH(R0)
POP(R0)
WRINT(R0)
POP(LP)
POP(BP)
DEALLOCATE(0)
RTN

f:

PUSH(LP)
PUSH(BP)
CMOVE(LP,BP)
ALLOCATE(2)

|x=1

CMOVE(1,R0)
PUSH(R0)
POP(R0)
PUTFRAME(R0,0)

CMOVE(1,R0)
PUSH(R0)
POP(R0)
PUTFRAME(R0,4)
GETFRAME(0,R0)
PUSH(R0)
GETFRAME(4,R0)
PUSH(R0)
GETFRAME(-12,R0)
PUSH(R0)
POP(R2)
POP(R1)
ADD(R2,R1,R0)
PUSH(R0)
POP(R0)
ST(R0,a)
POP(LP)
POP(BP)
DEALLOCATE(2)
RTN()

debut

CALL(main)
HALT()
```

```
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (2)
CMOVE(1, r0)
PUSH (r0)
POP (r0)
PUTFRAME(r0, 1)
CMOVE(1, r0)
PUSH (r0)
POP (r0)
PUTFRAME(r0, 5)
GETFRAME(r0, 1)
PUSH (r0)
GETFRAME(r0, 1)
PUSH (r0)
GETFRAME(r0, 5)
PUSH (r0)
POP (r2)
POP (r1)
ADD (r1, r2, r3)
PUSH (r3)
POP (r2)
POP (r1)
ADD (r1, r2, r3)
PUSH (r3)
POP (r0)
ST(r0, a)
DEALLOCATE (2)
POP (BP)
POP (LP)
RTN()

main:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
ALLOCATE (0)
CMOVE(3, r0)
PUSH (r0)
CALL(f)
DEALLOCATE(1)
LD(a, r0)
PUSH (r0)
POP (r0)
WRINT()
DEALLOCATE (0)
POP (BP)
POP (LP)
RTN()

debut:
CALL(main)
HALT()
pile:
 nom : f; type : void; categorie : fonction; nbparam : 1; nbloc : 2;
 nom : main; type : void; categorie : fonction;
 nom : a; type : int; categorie : global; valeur : 10;
 nom : i; type : int; categorie : param; valeur : ; rang : 0; scope : f;
 nom : x; type : int; categorie : local; valeur : ; rang : 0; scope : f;
 nom : y; type : int; categorie : local; valeur : ; rang : 1; scope : f;
```

Arbre 8:

```
CMOVE(pile, SP)
BR(debut)

|data

a : LONG(10)

|instructions
```

le code généré ne correspond pas.

```
nom : f; type : void; categorie : fonction; nbparam : 2; nbloc : 1;
nom : main; type : void; categorie : fonction;
nom : a; type : int; categorie : global; valeur : 10;
nom : x; type : int; categorie : local; valeur : ; rang : 0; scope : f;
nom : i; type : int; categorie : param; valeur : ; rang : 0; scope : f;
nom : j; type : int; categorie : param; valeur : ; rang : 1; scope : f;
```

```
main :
PUSH(LP)
PUSH(BP)
MOVE(SP,BP)
ALLOCATE(0)
CMOVE(1, R0)
PUSH(R0)
CMOVE(2,R0)
PUSH(R0)
CALL(f)
DEALLOCATE(2)
POP(R0)
ST(R0,a)
LD(a,R0)
PUSH(R0)
POP(R0)
WRINT()
BR(ret_main)

ret_main :
DEALLOCATE(0)
POP(BP)
POP(LP)
RTN()

f :

PUSH (LP)
PUSH(BP)
MOVE(SP,BP)
ALLOCATE(1)
GETFRAME(-16,R0)
PUSH(R0)
GETFRAME(-12,R0)
PUSH(R0)
POP(R2)
POP(R1)
ADD(R1,R2,R0)
PUSH(R0)
POP(R0)
PUTFRAME(RO, -20)
BR(ret_f)

ret_f:

DEALLOCATE(1)
POP (BP)
POP(LP)
RTN()

debut :

CALL(main)
HALT()
```

Exemples de programmes sources écrits dans notre langage :

Exemple 1 :

```
fonction void main () []
```

Exemple 2 :

```
$i :: 10;
$j :: 20;
$k;
$l;
fonction void main () []
```

Exemple 3 :

```
$i :: 10;
$j :: 20;
$k;
$l;
fonction void main () [
        $k::2
        $l::$i+3.$j
]
```

Exemple 4 :

```
$i;
$j :: 20;
fonction void main () [
        $i::lire()
        ecrire($i+$j)
]
```

Exemple 5 :

```
$i;
fonction void main () [
        $i::lire()
        Si(i>10)[
                ecrire(1)
                ecrire(2)
        ]
]
```

Exemple 6 :

```
$i;
$n :: 5;
```

```
fonction void main () [
	$i::0
	TantQue($i<$n)[
		ecrire($i)
		$i::$i+1
	]
]
```

Exemple 7 :

```
$a::10
fonction void f ($i) [
	$x, $y
	$x::1
	$y::1
	$a::$i+$x+$y
]
fonction void main () [
	f(3)
	ecrire($a)
]
```

Exemple 8 :

```
$a
fonction void f ($i, $j) [
	$x
	$x::$i+$j
	retourne($x+10)
]
fonction void main () [
	$a::f(1,2)
	ecrire($a)
]
```

Bilan critique :

Le projet est inachevé. Cela est dû à une mauvaise gestion du temps et de nombreux problèmes d'organisations et de compréhensions du sujet. Une grande partie de ces problèmes venaient aussi d'une connaissance faible dans le langage assembleur beta et nous avons perdu beaucoup de temps à corriger des erreurs liées à cela.

Conclusion :

De nombreux ralentissements dus à une mauvaise compréhension de certains points du projet nous ont empêché d'aller jusqu'au bout de celui-ci. Malgré cela nous pensons avoir tout de même appris certaines connaissances sur le fonctionnement et la construction d'un compilateur dans un contexte plus général. Enfin, nous avons aussi acquis plus d'expérience en travail de groupe ce qui est aussi un point important pour la suite de nos études. En l'occurrence le début du projet ayant mal démarré nous avons dû faire de nombreux changement par la suite qui nous a pris beaucoup de temps et nous avons dû faire des sacrifices au niveau du temps que nous avons investi dans le projet.