ALF

Génération de code

Bibliographie pour aujourd'hui

Keith Cooper, Linda Torczon, *Engineering a Compiler*

- Chapitre 5
  - 5.1
  - 5.2
  - 5.3

Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools (2nd Edition)*

- Chapitre 6
  - 6.1
  - 6.2
  - 6.3
  - 6.4
  - 6.5

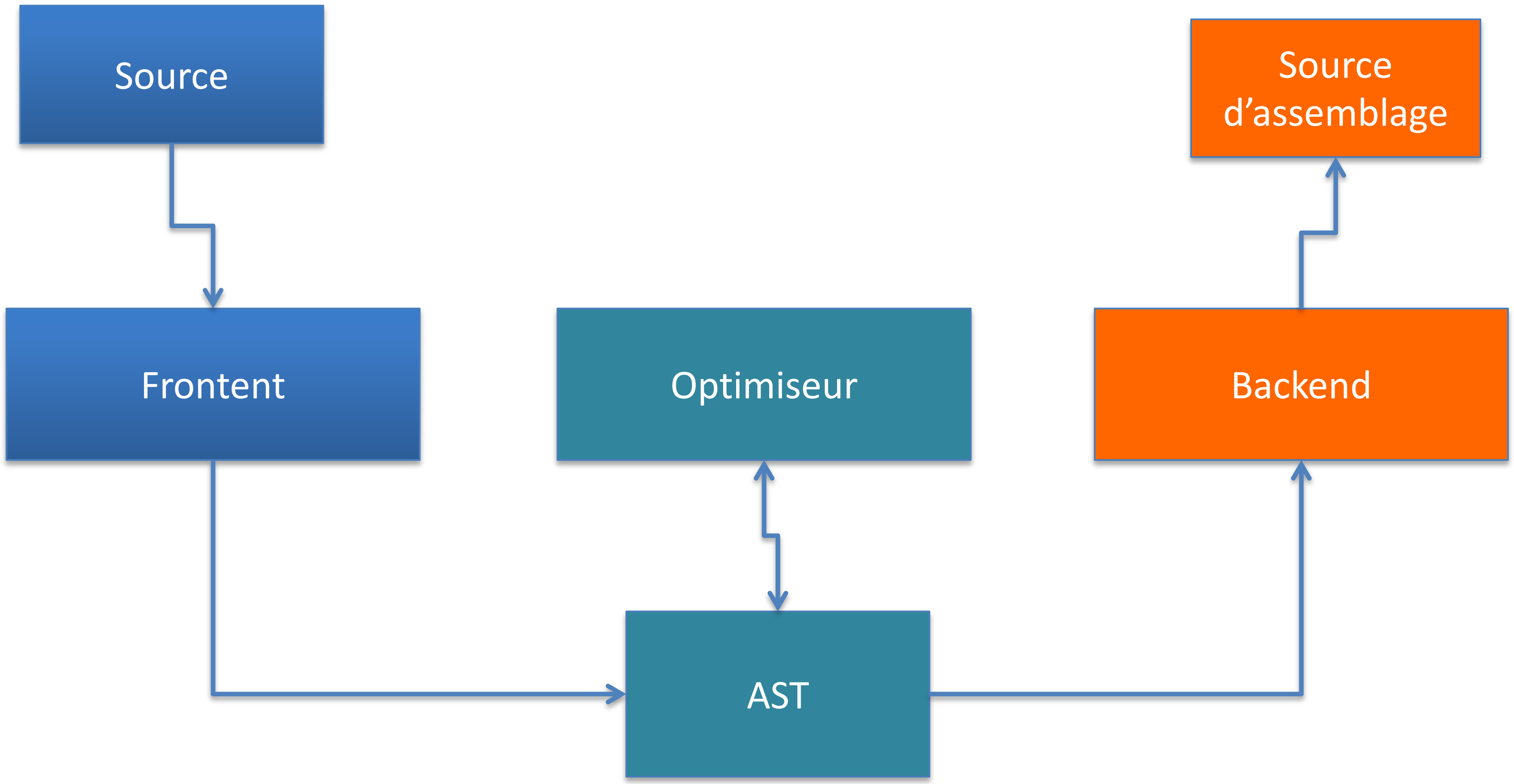
Contenu

- Three Address Code
  - évaluation des expression
  - contrôle de flux
    - branche
    - boucle
  - fonction
- Single Static Assignment

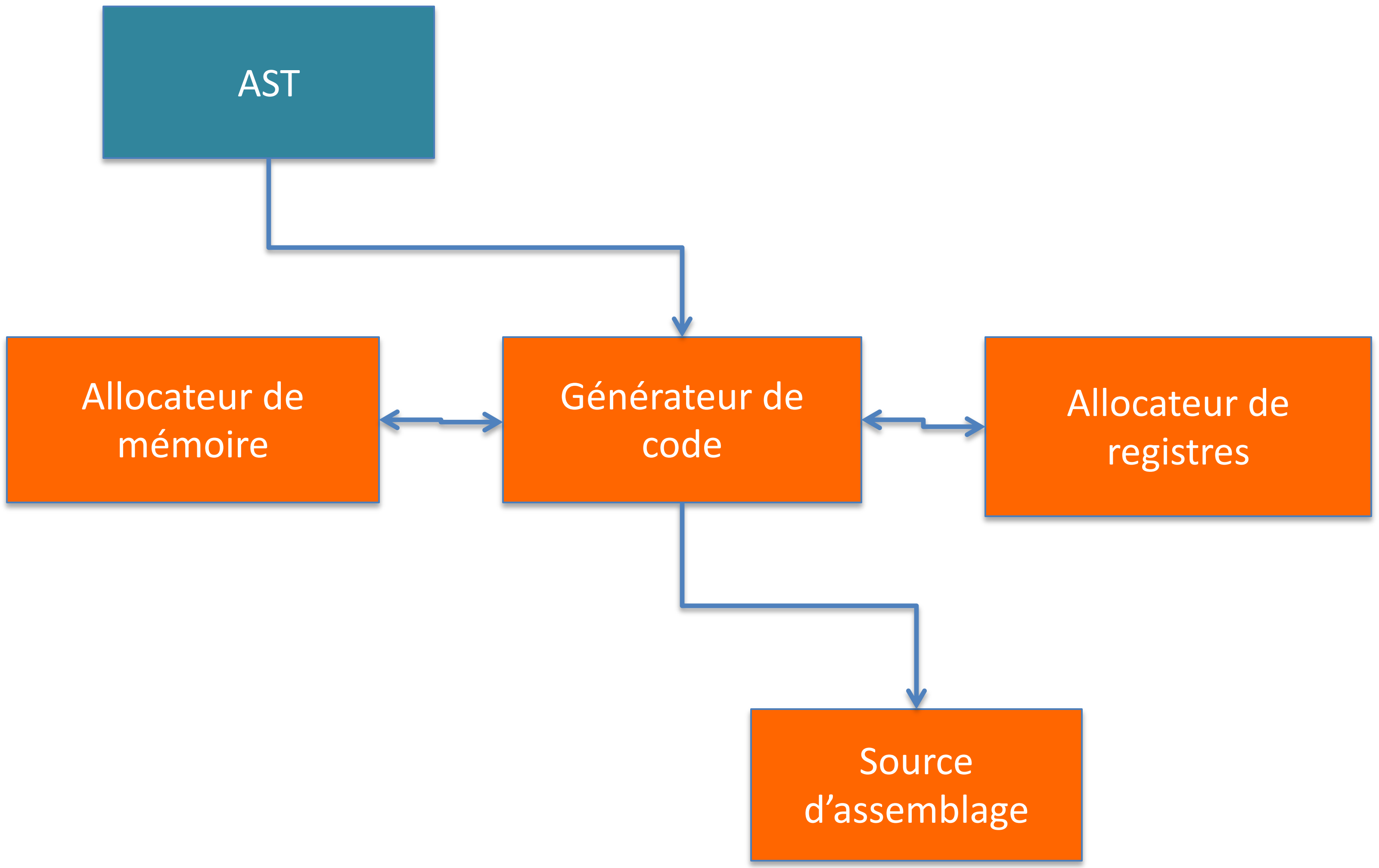
Edsger Wybe Dijkstra

- Néerlandais
- Leiden University
- Dijkstra Algorithm
- ALGOL 60
- Sémaphore
- Programmation Structuré
- Programmation Multithreaded

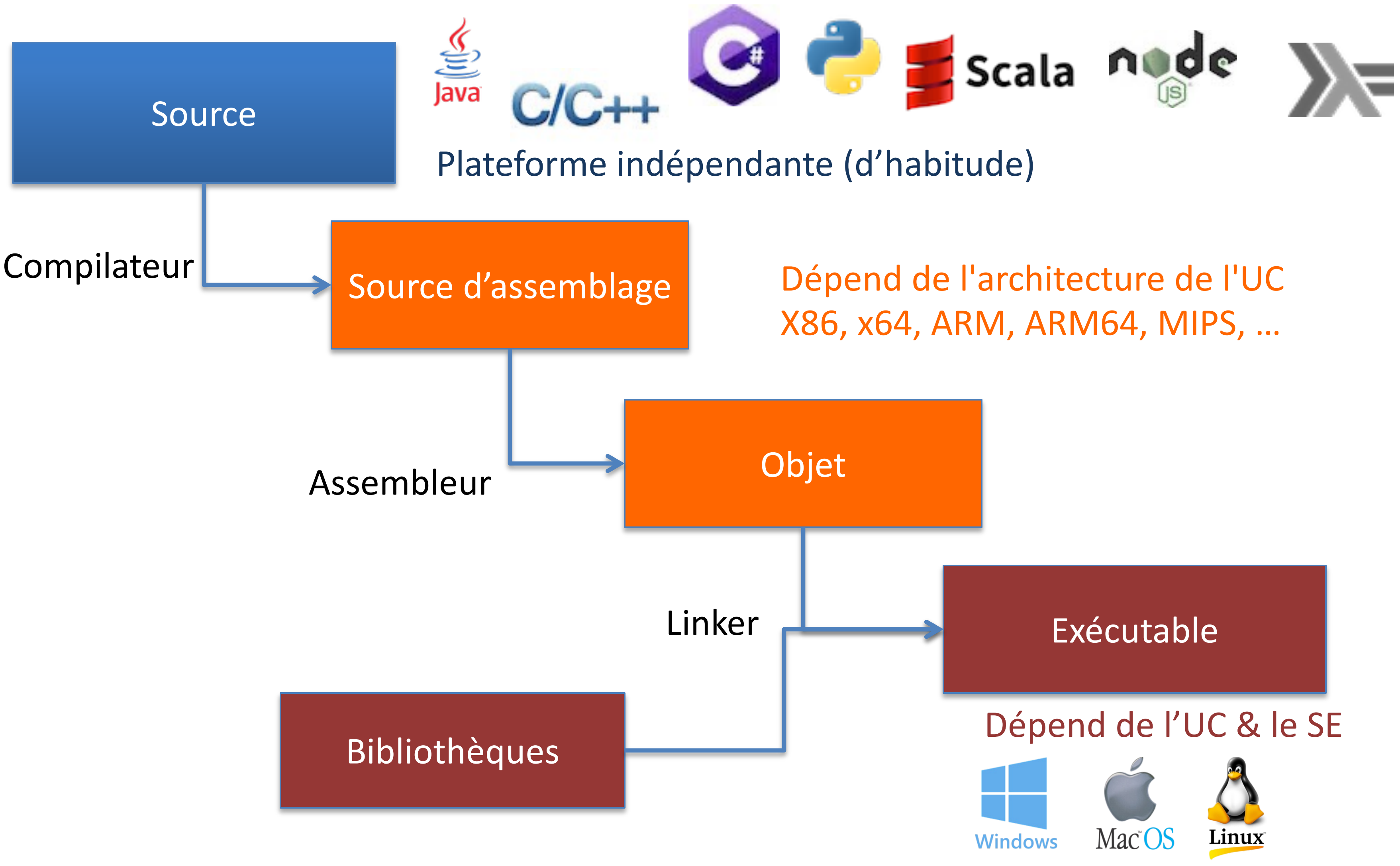
Pièces de compilation



Backend


AST
Allocateur de mémoire
Générateur de code
Allocateur de registres
Source d'assemblage

Compilateur



Three Address Code

- Instructions contenant 3 adresses
  - les opérandes
  - le résultat
- Un seul operateur

Type des instructions

- Mathématique
- Copie
- Saut inconditionnel (jump)
- Saut conditionnel (jump)
  - Simple
  - Avec condition
- Appel de fonction
- Copie indexée
- Assignement du pointeur

Enregistre le three address code

- result
- arg1
- arg2
- op

result	arg1	arg2	op
t1	a	b	+
t2	a		-
t3	a	b	+
t4	t1	t2	-
t3	s	t4	+

Mathématique

r = x op y

r = op y

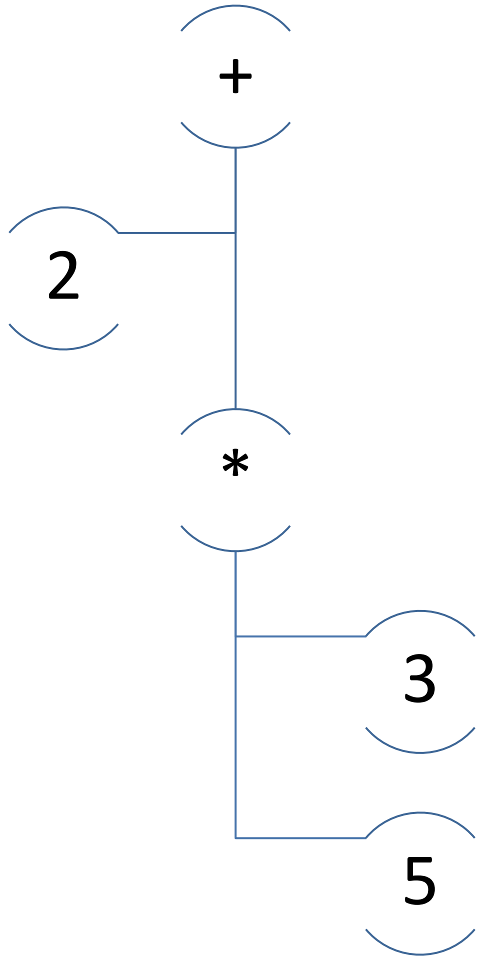
op: + - * / %

== <= >= < >

Exercices

- 2+3*5
- (6-2)*4
- 10 /5 + 2*3
- 3- (-2) *6
- -10/2 – (2+4)/2*(7-(-1))

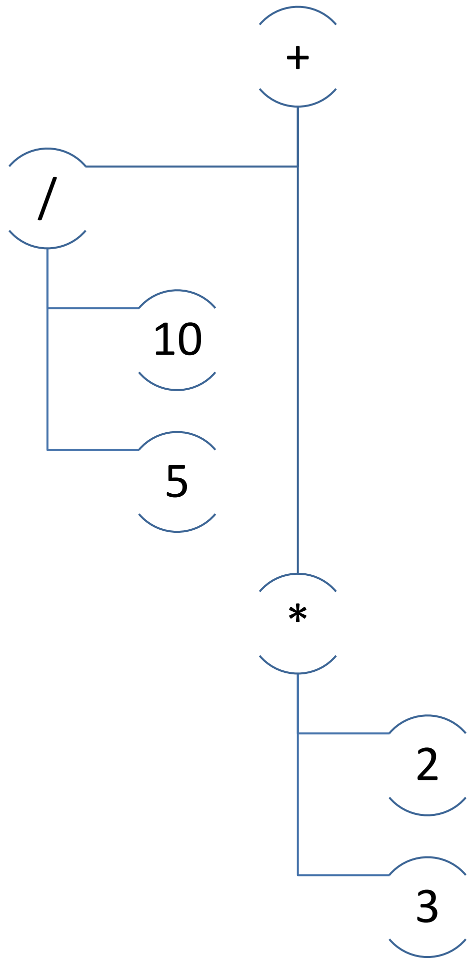
Exercices (2+3*5)



t1 = 3 * 5

t2 = 2 * t

Exercices (10 /5 + 2*3)



t1 = 10 / 5

t2 = 2 * 3

t3 = t1 + t2

Copie

$$x = y$$

Saut inconditionnel

- goto name
- label name

```
goto next
x = 2 + 3 ; this is jumped
label next
```

Saut conditionnel

- if x goto name
- ifFalse x goto namefalse
- label name

```
if f next
x = 2 + 3 ; this is jumped if
f is true
label next
```

Exercises

```
if (x+y > 3)
{
        a = 11;
}
```

Exercises

```
if (x+y > 3)
{
        a = 11;
}
```

Exemple

```
if (x+y > 3)
{
        a = 11;
}
```

```
t1 = x + y
t2 = t1 > 3
ifFase t2 goto endif
a = 11
label endif
```

Exercises

```
if (x+y > 3)
{
        a = 11;
}
else
{
        a = 12;
}
```

Exemple

```
if (x+y > 3)
{
        a = 11;
}
else
{
        a = 12;
}
```

```
t1 = x + y
t2 = t1 > 3
if t2 goto then
a = 12
goto endif
label then
a = 11
label endif
```

Exercises

```
if (x+y > 3 && y < x+90)
{
        a = 11;
}
else
{
        a = 12;
}
```

Exercises

```
if (x+y > 3 && y < x+90)
{
        a = 11;
}
else
{
        a = 12;
}
```

```
t1 = x + y
t2 = t1 > 3
t3 = x + 90
t4 = y < t3
t5 = t2 && t4
if t5 goto then
a = 12
goto endif
label then
a = 11
label endif
```

Exercises

```
while (x > 3)
{
        x = x + 1;
}
```

Exercises

```
while (x > 3)
{
        x = x + 1;
}
```

```
label while
t1 = tx > 3
ifFalse t1 goto endwhile
x = x + 1
goto while
label endwhile
```

Exercises

```
do
{
        x = x + 1;
} while (x+y > 3 && y < x+90);
```

Exercises

```
do
{
        x = x + 1;
} while (x+y > 3 && y < x+90);
```

```
label do
x = x + 1
t1 = x + y
t2 = t1 > 3
t3 = x + 90
t4 = y < t3
t5 = t2 && t4
if t5 goto do
```

Exercises

```
for (x=1; x + y > 3; x = x + 1)
{
        y = y + 7;
}
```

Exercises

```
for (x=1; x + y > 3; x = x + 1)
{
        y = y + 7;
}
```

```
x = 1
label for
y = y + 7
x = x + 1
t1 = x + y
t2 = t1 > 3
if t2 goto for
```

Appel de fonction

- param parameter
- call f, n
- r = call f, n

```
param a
param n
r = call power, 2
```

Exercises

```
void print (int x, int y)
{
        printf (“%s”, x);
        printf (“%s”, y);
}

print (2, 4);
```

Exercises

```
void print (int x, int y)
{
        printf ("%u", x);
        printf ("%u", y);
}

print (2, 4);
```

```
label start

label print
param "%u"
param x
call printf, 2
param "%u"
param y
call printf, 2
return

start:
param 2
param 4
call print, 2
```

Exercises

```
int expression (int x, int y, int z)
{
        return x*(y+z);
}

expression (1, 2, 5);
```

Exercises

```
int expression (int x, int y, int z)
{
	return x*(y+z);
}

expression (1, 2, 5);
```

```
goto start

label expression
n1 = y+z
n2 = n1*x
return n2

label start
param 1
param 2
param 5
call expression, 3
```

Exercises

```
int expression (int x, int y, int z)
{
        return x*(y+z);
}

expression (2+3, a+2*6, f(3));
```

Exercises

```
int expression (int x, int y, int z)
{
        return x*(y+z);
}

expression (2+3, a+2*6, f(3));
```

```
goto start

label expression
n1 = y+z
n2 = n1*x
return n2

label start
m1 = 2+3
param m1
m2 = 2*6
m3 = a+m2
param m3
param 3
m4 = call f,1
param m4
call expression, 3
```

Copie indexée

- x = a[i]
- x = a.element

```
x = a[i]
p = x.price
```

Exercises

a[5]

a[4][5]

a[i+j]

a.lst[i+j]

a.lst[i][j]

Assignement du pointeur

```
r = &x
```

Single Static Assignment

- Similaire avec Three Address Code
- Les variables sont des constantes
- Une fois attribuée, une variable ne peut pas changer sa valeur
- Fonction ϕ

Exemple

2*3+(5-3)

Three Address Code	**Single Static Assignment**
t=2 * 3	t1 = 2 * 3
t2=5 - 3	t2 = 5 – 3
t = t + t2	t3 = t1 + t2

Fonction ϕ

Source

```
if (x+y > 3)
{
        a = 11;
}
else
{
        a = 12;
}
```

Single Static Assignment

```
t1 = x + y
t2 = t1 > 3
ifFalse t2 goto next
t3 = 11
goto endif
label next
t4 = 12
endif
t5 = ϕ (t3, t4)
```

Sujets

- Three Address Code
  - évaluation des expression
  - contrôle de flux
    - branche
    - boucle
  - fonction
- Single Static Assignment

Questions