Plan

- Représentation des nombres
- Circuits logiques
- Unité Arithmétique et Logique
- Notions de temps et de mémorisation
- Contrôle et jonction des composants
- Evolution des ordinateurs – Historique
- Un microprocesseur simple
- Programmation d'un microprocesseur
- Système complet
- Les microprocesseurs actuels
- Exploitation de la performance des microprocesseurs

Exploitation de la Performance

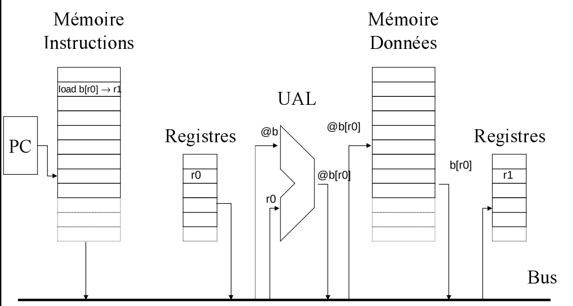
Programme source

```
for (i=0; i < n; i++) {
  a[i] = b[i] + c[i]
}
```

Programme assembleur

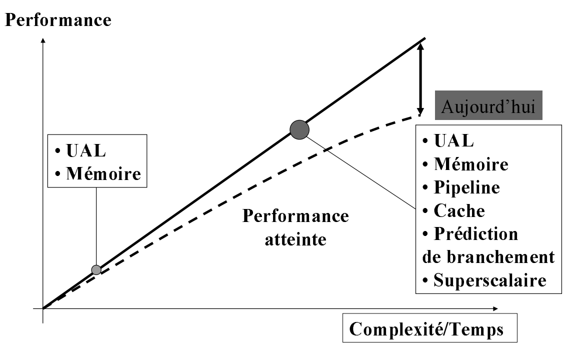
```
0 → r0                  ; i=0
label:
load b[r0] → r1         ; b[i]
load c[r0] → r2         ; c[i]
r1 + r2 → r3            ; r3 = b[i] + c[i]
store r3 → a[r0]        ; a[i] = r3
r0 + 1 → r0             ; i = i + 1
goto label if r0 < n    ; continuer si i<n
```

Etapes d'un Programme sur un Processeur

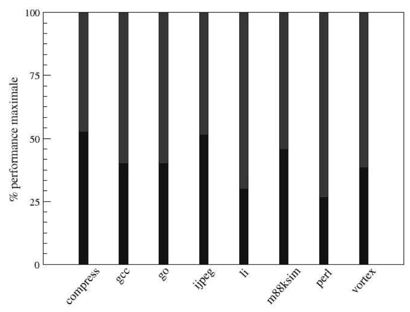


load b[r0] → r1

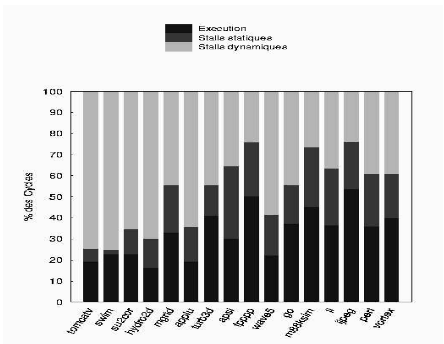
Où en Est-on ?



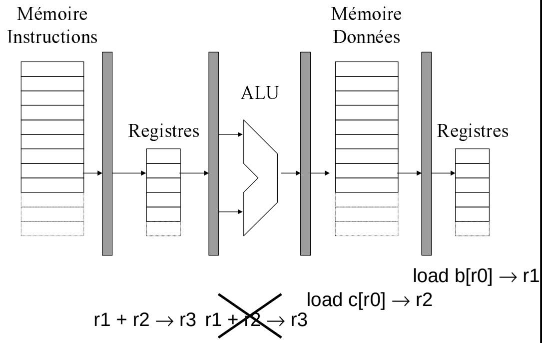
En Pratique…



Plus Précisément…



Augmenter le Débit des Instructions



Augmenter le Débit des Instructions

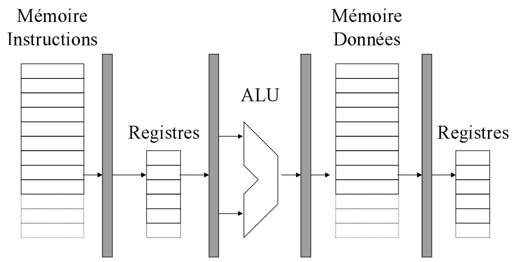
```
void test(int n) {
  for (i=0; i < n; i+=2) {
    a[i] = c[i] + b[i]
    a[i+1]=c[i+1]+b[i+1]
  }
}
```

Programme source

```
.........
load b[r0]   → r1
load b[r0+1] → r11
load c[r0]   → r2
load c[r0+1] → r12
r1 + r2 → r3
r11 + r12 → r13
.........
```

Programme assembleur

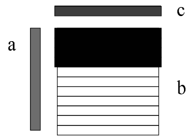
Augmenter le Débit des Instructions

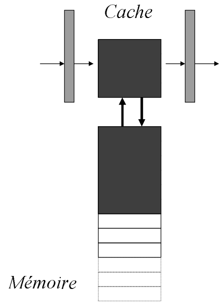


- Optimisations à la compilation (*unrolling*, *software pipelining…*).

Masquer la Distance à la Mémoire

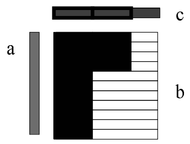
```
for (j=0; j < n; j++) {
  for (i=0; i < n; i++) {
    a[j] += b[j][i] + c[i];
  }
}
```

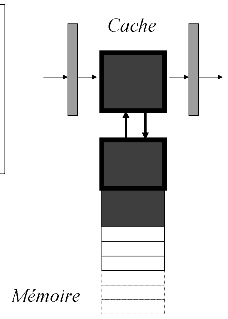




Masquer la Distance à la Mémoire

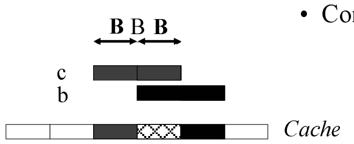
```
for (ii=0; ii < n; ii += B) {
  for (j=0; j < n; j++) {
    for (i=ii; i < ii+B; i++) {
      a[j] += b[j][i] + c[i];
…
```



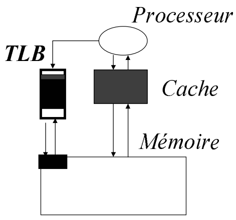


Masquer la Distance à la Mémoire

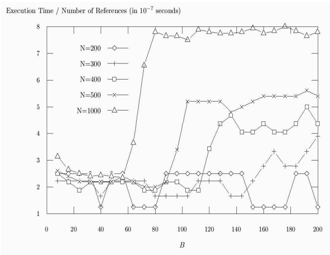
- Conflits de cache.




```
for (jj=0; jj < n; jj += B) {
  for (ii=0; ii < n; ii += B) {
    for (j=jj; j < jj+B; j++) {
      for (i=ii; i < ii+B; i++) {
        a[j] += b[j][i] + c[i];
…
```



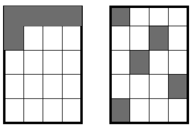
En Pratique…

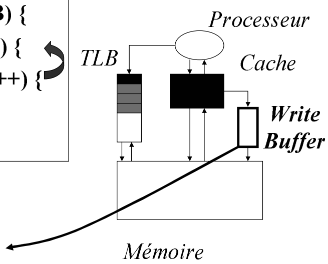


Transposée de Matrices NxN

Masquer la Distance à la Mémoire

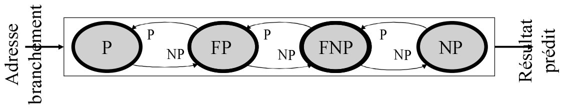
```
for (jj=0; jj < n; jj += B) {
 for (ii=0; ii < n; ii += B) {
  for (i=ii; i < ii+B; i++) {
   for (j=jj; j < jj+B; j++) {
    a[i][j] = b[j][i];
…
```





Anticiper le Comportement du Programme

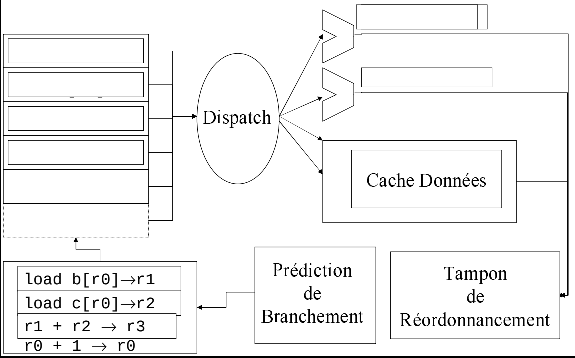
- Prédire le résultat du branchement.



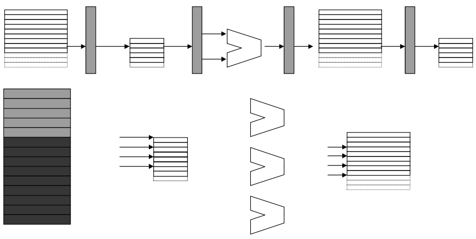

```
…if (b[i] > 5) {
   a[i] = b[i] + c[i];
  }
```

```
test (n=1);
…
test (n=4);
…
void test(int n) {
for (i=0; i < n; i++) {
```

Aujourd'hui: **Processeur** Superscalaire = **Exécuter les Instructions en Parallèle**

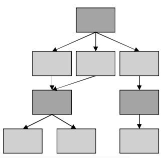


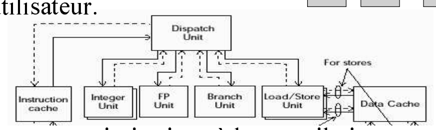
Les Mêmes Limitations…Décuplées + Trouver du Parallélisme



Comment Aider le Programmeur ?

- Arbre de décision: phases d'**analyse** et d'**optimisation**.
- Exhiber le comportement du programme sur l'architecture à l'utilisateur.





- A terme: optimisations à la compilation.
- Anticiper les architectures pour anticiper les optimisations nécessaires (simulateurs)

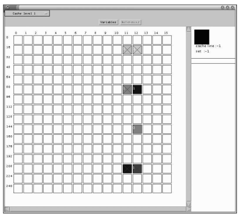
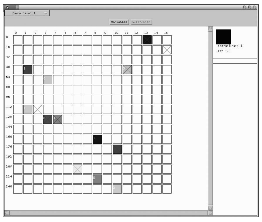
Un Cas Réel

• SpecFP95/00 SWIM (météo): 228s sur Alpha.

L1: 57%, L2: 5%, L3: 58%, TLB: 0,19%.

```
…
DO 200 J=1,N
 DO 200 I=1,M
   UNEW(I+1,J) = UOLD(I+1,J)+
        .    TDTS8*(Z(I+1,J+1)+Z(I+1,J))*(CV(I+1,J+1)
        .   +CV(I,J+1)+CV(I,J)
        .   +CV(I+1,J))-TDTSDX*(H(I+1,J)-H(I,J))
   VNEW(I,J+1) = VOLD(I,J+1)-TDTS8*(Z(I+1,J+1)+Z(I,J+1))
        .       *(CU(I+1,J+1)+CU(I,J+1)+CU(I,J)+CU(I+1,J))
        .       -TDTSDY*(H(I,J+1)-H(I,J))
   PNEW(I,J) = POLD(I,J)-TDTSDX*(CU(I+1,J)-CU(I,J))
   1       -TDTSDY*(CV(I,J+1)-CV(I,J))
200 CONTINUE
…
```

Cas Réel (*padding*)

```
COMMON U(N1,N2), V(N1,N2),…
COMMON U(N1,N2), PAD1(71), V(N1,N2),…
```

228s → 139s

Cas Réel (*blocking*)

```
DO II=1,M,BI
   DO 200 J=1,N
      DO 200 I=1,MIN(M,II+BI-1)
         …CV(I,J+1)…CV(I,J)…
```

139s → 171s

```
DO 200 JJ = 1,N, BJ
   DO 200 II=1, M, BI
      DO 200 J = JJ, MIN(N,JJ+BJ-1)
         DO 200 I = II, MIN(M,II+BI-1)
```

139s → 136s

Cas Réel (fusion de boucles)

```
DO 10 j = 1, n
  DO 10 i = 1, m
    ... = u(i,j)
10 CONTINUE
```

```
DO 20 j = 1, n
  DO 20 i = 1, m
    u(i,j) = ...
20 CONTINUE
```

```
DO 10 j = 1, n
  DO 10 i = 1, m
    ... = u(i,j)
    u(i,j) = ...
10 CONTINUE
```

136s → 127s

Cas Réel (fusion de boucles)

Problèmes de Légalité

```
DO 10 j = 1, n
  DO 10 i = 1, m
    ... = u(i-1,j)
10 CONTINUE
DO 20 j = 1, n
  DO 20 i = 1, m
    u(i,j) = ...
20 CONTINUE
```

```
DO 10 j = 1, n
  DO 10 i = 1, m
    ... = u(i-1,j)
    u(i,j) = ...
10 CONTINUE
```

Cas Réel (fusion de boucles)

Problèmes de Légalité

```
DO 10 j = 1, n
  DO 10 i = 1, m
    ... = u(i-1,j)
10 CONTINUE
```

```
DO 10 j = 1, n
  DO 10 i = 0, m-1
    ... = u(i,j)
10 CONTINUE
```

```
DO 9 j = 1, n
  DO 9 i = 0, 0
    ... = u(i,j)
9 CONTINUE
DO 10 j = 1, n
  DO 10 i = 1, m-1
    ... = u(i,j)
10 CONTINUE
```

&

```
DO 20 j = 1, n
  DO 20 i = 1, m-1
    u(i,j) = ...
20 CONTINUE
DO 11 j = 1, n
  DO 11 i = m, m
    u(i,j) = ...
11 CONTINUE
```

Cas Réel (fusion de boucles)

Problèmes de Légalité

```
DO 10 j = 1, n
   DO 10 i = 1, m
      ... = u(i-1,j)
10 CONTINUE
DO 20 j = 1, n
   DO 20 i = 1, m
      u(i,j) = ...
20 CONTINUE
```

→

```
DO 9 j = 1, n
   DO 9 i = 0, 0
      ... = u(i,j)
9 CONTINUE
DO 10 j = 1, n
   DO 10 i = 1, m-1
      ... = u(i,j)
      u(i,j) = ...
10 CONTINUE
DO 11 j = 1, n
   DO 11 i = m, m
      u(i,j) = ...
11 CONTINUE
```

Cas Réel (substitution)

```
DO 10 i
   DO 10 j
      CU(i,j) = 0.5*(P(i,j-1)+P(i-1,j))*U(i,j)
10 CONTINUE
DO 20 i
   DO 20 j
      ... = CU(i,j) + CU(i,j+1)
20 CONTINUE
```

```
DO 20 i
   DO 20 j
      ... = 0.5*(P(i,j-1)+P(i-1,j))*U(i,j)
      + 0.5*(P(i,j)+P(i-1,j+1))*U(i,j+1)
20 CONTINUE
```

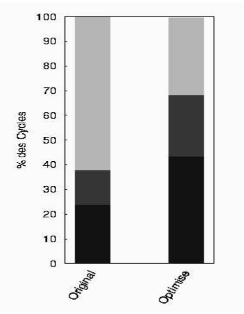
- 60% de calculs en plus

127s → 97s

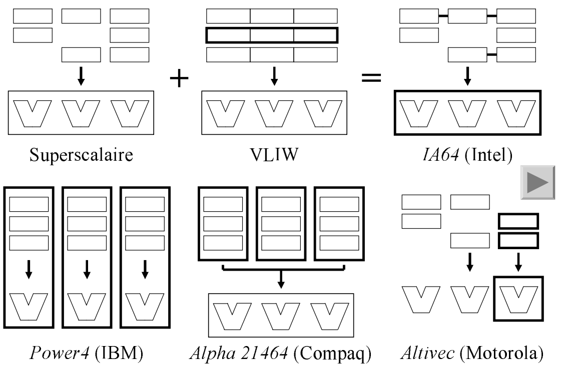
Cas Réel

Bilan

228s → 97s (speedup = 2,35)



Où Va-t-on ?



Où Va-t-on ?

Rechercher des architectures dont il soit plus facile d'exploiter la performance potentielle.