

Exercices - Semaine XIII - Solutions

Question 1:

a)

1. sub	r1, r3, r2		r2 <- r1 - r3
2. and	r2, r5, r12		r12 <- r2 and r5
3. or	r6, r2, r13		r13 <- r6 or r2
4. add	r2, r2, r14		r14 <- r2 + r2
5. move	r2, r15		r15 <- r2

Dépendance de type RAW entre I1 et I2 [r2]

Dépendance de type RAW entre I1 et I3 [r2]

Dépendance de type RAW entre I1 et I4 [r2]

Dépendance de type RAW entre I1 et I5 [r2]

Le compilateur ne peut rien faire (sauf ajouter des NOPs):

1. sub	r1, r3, r2		r2 <- r1 - r3
2. NOP			
3. NOP			
4. and	r2, r5, r12		r12 <- r2 and r5
5. or	r6, r2, r13		r13 <- r6 or r2
6. add	r2, r2, r14		r14 <- r2 + r2
7. move	r2, r15		r15 <- r2

Le *bypassing* peut aider en réduisant le nombre de NOP. Dans l'exemple de notre pipeline:

1. sub	r1, r3, r2		r2 <- r1 - r3
2. and	r2, r5, r12		r12 <- r2 and r5
3. NOP			
4. or	r6, r2, r13		r13 <- r6 or r2
5. add	r2, r2, r14		r14 <- r2 + r2
6. move	r2, r15		r15 <- r2

On peut aussi remarquer que les instructions 2, 3, 4 et 5 peuvent être exécutées en parallèle.

b)

1. sub	r1, r3, r2		r2 <- r1 - r3
2. and	r2, r5, r4		r4 <- r2 and r5
3. or	r2, r4, r4		r4 <- r2 or r4
4. add	r4, r2, r9		r9 <- r4 + r2

Dépendance de type RAW entre I1 et I2 [r2]

Dépendance de type RAW entre I1 et I3 [r2]

Dépendance de type RAW entre I1 et I4 [r2]

Dépendance de type RAW entre I2 et I3 [r4]

Dépendance de type WAW entre I2 et I3 [r4]

Dépendance de type RAW entre I3 et I4 [r4]

Le compilateur ne peut rien faire (sauf ajouter des NOPs).

1. sub	r1, r3, r2		r2 <- r1 - r3
2. NOP			
3. NOP			
4. and	r2, r5, r4		r4 <- r2 and r5
5. NOP			
6. NOP			
7. or	r2, r4, r4		r4 <- r2 or r4
8. NOP			
9. NOP			
10. add	r4, r2, r9		r9 <- r4 + r2

Le *bypassing* peut aider en éliminant les NOPs. Dans l'exemple de notre pipeline:

1. sub	r1, r3, r2		r2 <- r1 - r3
2. and	r2, r5, r4		r4 <- r2 and r5
3. NOP			
4. NOP			
5. or	r2, r4, r4		r4 <- r2 or r4
6. add	r4, r2, r9		r9 <- r4 + r2

On peut aussi remarquer qu'aucune des instructions ne peut être exécutée en parallèle.

c)

1. add r4, r5, r2		r2 <- r4 + r5
2. add r2, r5, r4		r4 <- r2 + r5
3. load 100(r2), r5		r5 <- M[r2+100]
4. add r4, r2, r3		r3 <- r4 + r2

Dépendance de type RAW entre I1 et I2 [r2]

Dépendance de type RAW entre I1 et I3 [r2]

Dépendance de type RAW entre I1 et I4 [r2]

Dépendance de type WAR entre I1 et I2 [r4]

Dépendance de type RAW entre I2 et I4 [r4]

Dépendance de type WAR entre I1 et I3 [r5]

Dépendance de type WAR entre I2 et I3 [r5]

Le compilateur ne peut rien faire (sauf ajouter des NOPs).

1. add r4, r5, r2		r2 <- r4 + r5
2. NOP		
3. NOP		
4. add r2, r5, r4		r4 <- r2 + r5
5. load 100(r2), r5		r5 <- M[r2+100]
6. add r4, r2, r3		r3 <- r4 + r2

Le *bypassing* peut aider en réduisant le nombre de NOP. Dans l'exemple de notre pipeline:

1. add r4, r5, r2		r2 <- r4 + r5
2. add r2, r5, r4		r4 <- r2 + r5
3. NOP		
4. load 100(r2), r5		r5 <- M[r2+100]
5. add r4, r2, r3		r3 <- r4 + r2

On peut aussi remarquer que les instructions 3 et 4 peuvent être exécutées en parallèle.

Question 2:

a) Si l'on déroule la boucle (deux itérations suffisent):

```
1a.  a[2] = b[2] + a[2];
2a.  c[1] = a[2] + d[2];
3a.  a[1] = 2 * b[2];
4a.  b[3] = 2 * b[2];
1b.  a[3] = b[3] + a[3];
2b.  c[2] = a[3] + d[3];
3b.  a[2] = 2 * b[3];
4b.  b[4] = 2 * b[3];
```

Dépendance de type RAW entre I1 et I2 (dans chaque itération) [a]
Les instructions 2, 3, 4 dans chaque itération sont parallélisables.

Dépendance entre itérations de type RAW entre I4a et I1b [b]

Dépendance entre itérations de type WAR entre I1a et I7b [a]

Dépendance entre itérations de type WAW entre I1a et I7b [a]

La boucle n'est PAS parallélisable.

b) Si l'on déroule la boucle (deux itérations suffisent):

```
1a.  a[1] = a[1] + b[1];
2a.  b[2] = c[1] + d[1];
1b.  a[2] = a[2] + b[2];
2b.  b[3] = c[2] + d[2];
```

Pas de dépendance à l'intérieur de chaque itération.

Les instructions 1 et 2 sont donc parallélisables.

Dépendance entre itérations de type RAW entre I2a et I1b [b]

La boucle n'est PAS parallélisable ... ou est-elle?

La boucle peut être transformée en:

```
a[1] = a[1] + b[1];
for (i=1; i<=99; i++) {
    b[i+1] = c[i] + d[i];
    a[i+1] = a[i+1] + b[i+1];
}
b[101] = c[100] + d[100];
```

Ce qui introduit une dépendance à l'intérieur des itérations, mais rend la boucle parallélisable.