UE INF2245
Devoir non surveillé
Introduction au parallélisme

Frédéric Raimbault

Ce travail personnel est à remettre sur l'ENT sous la forme d'un fichier rédigé sur un traitement de texte et formaté en PDF. Toute similitude détectée entre deux rendus sera sanctionnée par un zéro à la note de l'UE.

1 Parallélisme au niveau instruction (ILP)

La lecture du chapitre 4 de "Computer Organization and Design : the hardware/software interface", 5e édition, D. Patterson et J. Hennessy est conseillée.

Exercice 1 : (5 points)

On considère la séquence d'instructions suivantes, correspondant à la compilation d'une boucle de base DAXPY (utilisée dans le benchmark LINPACK) pour le calcul de $Y = a \times X + Y$ où X et Y sont des vecteurs de 64 réels (en double précision sur 64-bits) en mémoire (adressable par octet) et a une valeur réelle scalaire. Les registres d'entiers sont notés Rx et les registres de réels Fx.

```
1                LD    F0,a          ; load scalar a
2                ADD   R4,Rx,#512    ; last address to load
3  Loop:         LD    F2,0(Rx)      ; load X[i]
4                MUL   F2,F2,F0      ; a x X[i]
5                LD    F4,0(Ry)      ; load Y[i]
6                ADD   F4,F4,F2      ; a x X[i] + Y[i]
7                SD    F4,9(Ry)      ; store into Y[i]
8                ADD   Rx,Rx,#8      ; increment index to X
9                ADD   Ry,Ry,#8      ; increment index to Y
10               SUB   R20,R4,Rx     ; compute bound
11               BNZ   R20,loop      ; check if done
```

Chaque instruction nécessite 5 cycles [1] :

1. IF : chargement de l'instruction
2. ID : décodage de l'instruction et lecture des registres
3. EX : exécution ou calcul d'adresse
4. MEM : accès à la mémoire (en écriture ou lecture)
5. WB : écriture du résultat

Calculez le nombre de cycles nécessaires à l'exécution de la séquence d'instructions du DAXPY sur un processeur possédant à chaque question successive différentes caractéristiques. Dans la réponse à chaque question vous donnerez le contenu du pipeline avec un tableau à double entrée de la forme suivante (chaque ligne représente un cycle d'instruction, chaque colonne représente un étage de pipeline). Vous expliciterez les cas de dépendance entre les instructions et les bulles engendrées dans le pipeline.

IF	ID	EX	MEM	WB
`LD F0,a`				
`ADD R4,Rx,512`	`LD F0,a`			
`LD F2,0(Rx)`	`ADD R4,Rx,512`	`LD F0,a`		

1. Sans pipeline d'instruction.
   Le chemin de données du processeur est donné en figure 1.
2. Avec pipeline d'instruction et lecture anticipée. Le chemin de données du processeur avec pipeline est donné en figure 2.
   Le principe de la lecture anticipée (forwarding) est donné sur la figure 3. Grace au «forwarding» le calcul résultat d'un cycle peut être utilisé dès le cycle suivant dans les deux instructions suivantes.
3. Avec pipeline d'instruction, «forwarding» et avec prédiction de branchement. On suppose que l'unité de prédiction de branchement prédit que le branchement est toujours pris.
4. Avec pipeline d'instruction, «forwarding», prédiction de branchement et exécution super-scalaire dans l'ordre. On suppose que le processeur est capable d'exécuter en parallèle dans le pipeline une instruction de lecture ou écriture mémoire avec une instruction arithmétique ou de branchement.
5. Avec pipeline d'instruction, prédiction de branchement et exécution super-scalaire dans le désordre. On suppose que le processeur est capable d'analyser les dépendances entre 10 instructions consécutives et de les exécuter dans le désordre ; on ajoute qu'il est capable d'exécuter en même temps une opération sur les entiers et une opération sur les flottants.

1. Les 5 cycles sont de durée égale et toujours présents même si rien n'y est fait pour certaines instructions

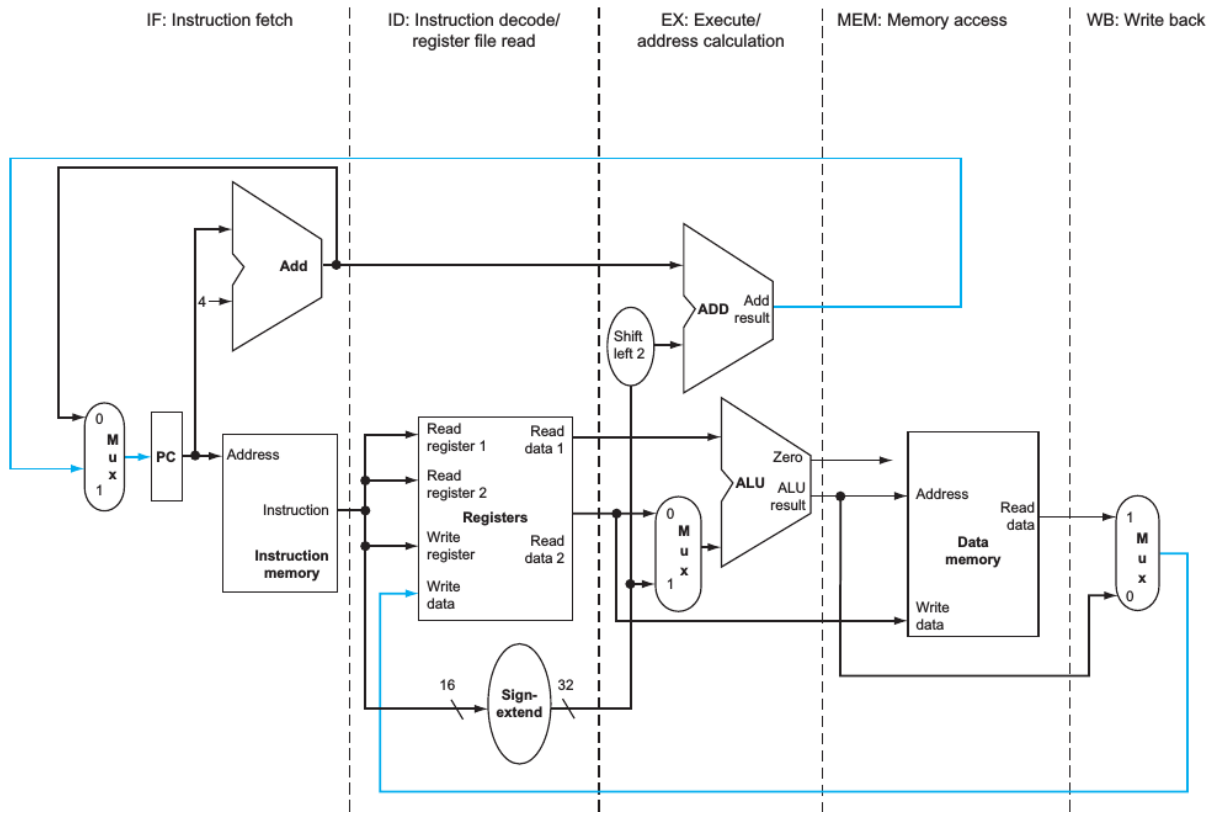

Fig. 1: Chemin de données du processeur sans pipeline

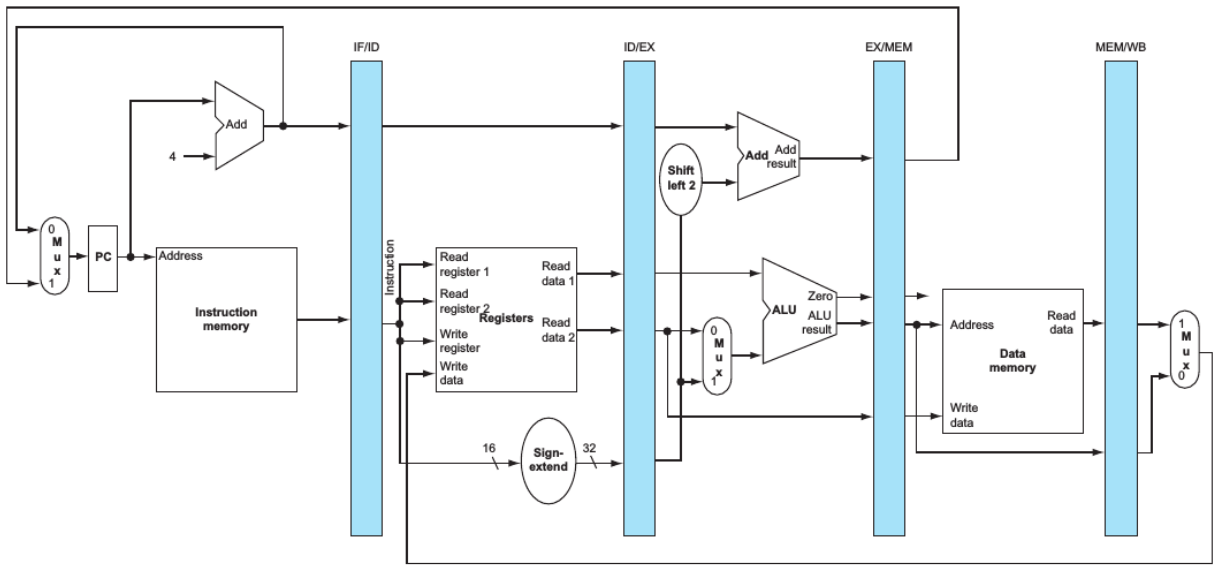

Fig. 2: Chemin de donnée du processeur avec pipeline

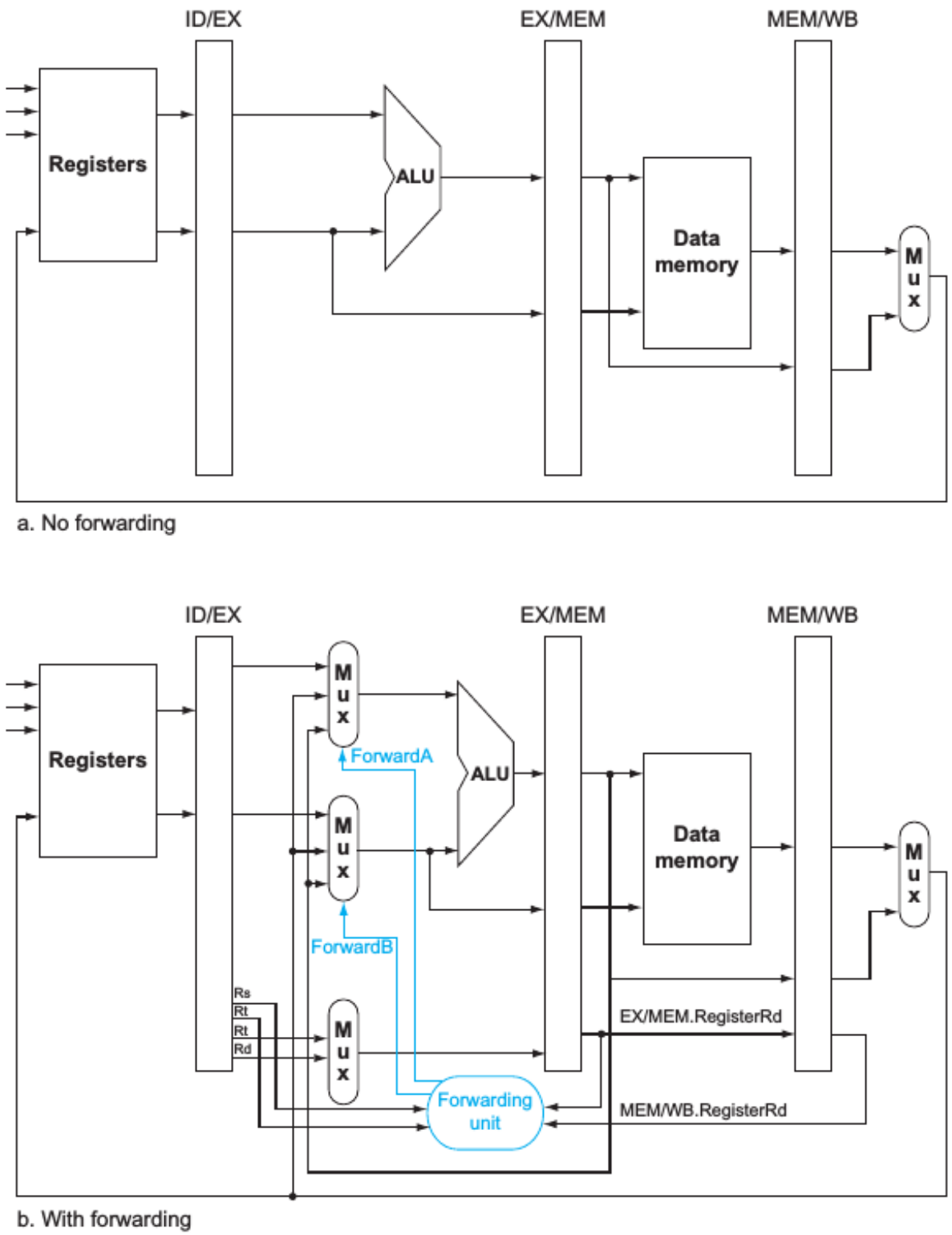

Fig. 3: Chemin de donnée du processeur avec pipeline et lecture anticipée

2 Loi d'Amdahl et calcul de performance

Les exercices suivants sont extraits de *"Computer Architecture, a quantitative approach", J. Hennessy and D. Patterson, page 76-100*

Exercice 2 : (1 point)

Suppose that we want to enhance the processor used for Web serving. The new processor is 10 times faster on computation in the Web serving application than the original processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement ?

Exercice 3 : (5 points)

In this exercise, assume that we are considering enhancing a machine by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is 10 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode the percentage of vectorization.

1. Draw a graph that plots the speedup as a percentage of the computation performed in vector mode. Label the y-axis "Net speedup" and label the x-axis "Percent vectorization."
2. What percentage of vectorization is needed to achieve a speedup of 2 ?
3. What percentage of the computation run time is spent in vector mode if a speedup of 2 is achieved ?
4. What percentage of vectorization is needed to achieve one-half the maximum speedup attainable from using vector mode ?
5. Suppose you have measured the percentage of vectorization of the program to be 70%. The hardware design group estimates it can speed up the vector hardware even more with significant additional investment. You wonder whether the compiler crew could increase the percentage of vectorization, instead. What percentage of vectorization would the compiler team need to achieve in order to equal an addition 2× speedup in the vector unit (beyond the initial 10×) ?

Exercice 4 : (4 points)

Your company has just bought a new Intel Core i5 dual-core processor, and you have been tasked with optimizing your software for this processor. You will run two applications on this dual core, but the resource requirements are not equal. The first application requires 80% of the resources, and the other only 20% of the resources. Assume that when you parallelize a portion of the program, the speedup for that portion is 2.

1. Given that 40% of the first application is parallelizable, how much speedup would you achieve with that application if run in isolation ?
2. Given that 99% of the second application is parallelizable, how much speedup would this application observe if run in isolation ?
3. Given that 40% of the first application is parallelizable, how much overall system speedup would you observe if you parallelized it ?
4. Given that 99% of the second application is parallelizable, how much overall system speedup would you observe if you parallelized it ?

Exercice 5 : (5 points)

When parallelizing an application, the ideal speedup is speeding up by the number of processors. This is limited by two things : percentage of the application that can be parallelized and the cost of communication. Amdahl's law takes into account the former but not the latter.

1. What is the speedup with N processors if 80% of the application is parallelizable, ignoring the cost of communication ?
2. What is the speedup with 8 processors if, for every processor added, the communication overhead is 0.5% of the original execution time.
3. What is the speedup with 8 processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time ?
4. What is the speedup with N processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time ?
5. Write the general equation that solves this question : What is the number of processors with the highest speedup in an application in which P% of the original execution time is parallelizable, and, for every time the number of processors is doubled, the communication is increased by 0.5% of the original execution time ?