

12207



By
Daniel
Dalati

2023

Course: Info 2207

Year: 2016-2017

Duration: 2 Hours

Review: 1^{time}Session

Exercise 1: (15 pts)

has. Consider a non-pipelined processor having a data path made up of modules whose execution times are respectively: 2, 3, 2, 4, 5, 2, 5, 2, 3 (in cycles). What is the cycle time of the unpipelined processor? Propose a pipelined version with 6 stages (not necessarily equal) and calculate the new cycle time knowing that the stabilization time between stages is 1 ns.

Unpipelined cycle time = $2+3+2+4+5+2+5+2+3=28$ cycles. A possible pipelined version: 5(2+3), 6(2+4), 5, 2, 5, 5(2+3).

Pipelined cycle time=maximum cycle time+latency
 $=6+1=7$ cycles

b. What is the register renaming technique? What types of conflicts can be resolved using this technique and which not? Justify.

Register renaming is a technique that aims to minimize the impact of EAL and EAE dependencies on instruction parallelism by dynamically assigning each value produced by a program to a new register, in order to break EAL and EAE dependencies.

La figure suivante illustre la manière dont le renommage de registres peut améliorer la performance.

Avant renommage	Après renommage
ADD r3, r4, r5	ADD hw3, hw4, hw5
LD r7, (r3)	LD hw7, (hw3)
SUB r3, r12, r11	SUB hw20, hw12, hw11
ST (r15), r3	ST (hw15), hw20

Exercise 2: (25 pts)

Consider a cache memory of size 32K bytes with a line size of 16 bytes, an associativity of order 8, managed in deferred writing and using an LRU type replacement policy. Main memory is byte addressable with 32-bit addresses.

has. Calculate the number of lines and sets in the cache.

$$\text{Nombre de lignes} = \frac{32 * 2^{10}}{16} = \frac{2^{15}}{2^4} = 2^{11} = 2048 \text{ lignes}$$

$$\text{Nombre d'ensembles} = \frac{2^{11}}{2^3} = 2^8 = 256 \text{ lignes}$$

1

- b. Calculer le nombre de bits de l'étiquette.

$$\text{Largeur de bus d'ad} = 32 = \text{nbBits.} \text{étiquette} + \text{nBits.} \text{ensemble} + \text{nbBits.} \text{Octet}$$

$$32 = \text{nbBits.} \text{étiquette} + \log_2 256 + \log_2 16$$

$$32 = \text{nbBits.} \text{étiquette} + 8 + 4$$

$$32 = \text{nbBits.} \text{étiquette} + 12$$

$$\text{nbBits.} \text{étiquette} = 32 - 12 = 20 \text{ Bits}$$

- C. For the following hexadecimal memory addresses, give the set number, the byte number referenced in the line and the label bits: 0xAB1DCEFF, 0xEFA00AC0. Subsequently give an example of two memory addresses which will be mapped onto the same cache set.

0xAB1DCEFF							
HAS	B	1	D	VS	E	F	F
Label					Together		Byte
1010	1011	0001	1101	1100	1110	1111	1111
		AB1DC			EF		F
		AB1DC			239		15

0xEFA00AC0							
E	F	HAS	0	0	HAS	VS	0
Label					Together		Byte
1110	1111	1010	0000	0000	1010	1100	0000
		EFA00			AC		0
		EFA00			172		0

Two memory addresses mapped to the same set (set 0):

Memory address	Label	Together	Byte
0x012AB000	012AB	00	0
0x8CFD100F	8CFD1	00	F

d. Donner la taille d'une entrée d'étiquette et celle du tableau d'étiquettes.

$$T.l'.etiquette = 1bit(Valid) + 1(ecr. différent) + nbBits.etique + nbBits.algo LRU$$

$$T.l'.etiquette = 1 + 1 + 20 + \log_2 8$$

$$T.l'.etiquette = 1 + 1 + 20 + 3 = 25 \text{ bits}$$

$$\text{Taille du tableau d'etiquettes} = 25 * 2^{11} = 50 * 2^{10} \text{ Bits} = 50 \text{ KBits}$$

e. Sachant qu'on garde la même taille de cache et de ligne de cache, comment faut-il modifier le reste des paramètres du cache (associativité et politique de mise à jour de la mémoire centrale) pour réduire au maximum la taille du tableau d'étiquettes, donner la nouvelle taille du tableau d'étiquettes.

Pour reduire au maximum la taille du tableau d'étiquettes il faut penser à réduire la taille de l'entrée du tableau d'étiquette.

2

• •

Pour arriver à faire cette réduction, on doit d'abord changer la politique d'écriture (utiliser l'écriture immédiate) et réduire au maximum le nombre de lignes par ensemble c-à-dire une ligne par ensemble.

Lorsqu'on réduit le nombre de lignes par ensemble on aura une modification au niveau de taille de l'étiquette qu'on doit recalculer.

$$\text{Largeur de bus d'ad} = 32 = nbBits.etiquette + nBits.index + nbBits.Octet$$

$$32 = nbBits.etiquette + \log_2 nbLignes + \log_2 16$$

$$32 = nbBits.etiquette + \log_2 2048 + 4$$

$$32 = nbBits.etiquette + 11 + 4$$

$$nbBits.etiquette = 32 - 15 = 17 \text{ Bits}$$

Donc :

$$T.l'.etiquette = 1bit(Validité) + nbBits.etique + nbBits.algo LRU$$

$$T.l'.etiquette = 1 + 17 + \log_2 1$$

$$T.l'.etiquette = 18 \text{ Bits}$$

$$\text{Taille du tableau d'etiquettes} = 18 * 2^{11} = 36 * 2^{10} \text{ Bits} = 36 \text{ KBits}$$

Exercise 3: (25 pts)

Consider a 1 KB direct-match cache, with lines (blocks) of 16 bytes. The processor has 32-bit registers and 32-bit addresses. The cache is rewritten (write back) and write allocated (in the event of a write failure, the block is brought into cache memory to write it).

has. What are the number of bits needed for the index, label and move (address in the row)?

1 KB = 2^{10} .

16-byte lines: 2⁴.

Number of lines: $2^6 = 64$.

Displacement: 4 bits

Index: 6 bits

Label: 22 bits

b. Consider the following C code:

```
int A[512], i ;  
for (i = 0; i < 256; i++) {  
  
    A[i] = A[i] + 1; A[i+256] = A[i+256] - 1;  
  
}
```

i. What is the failure rate (number of faults / number of hits) when running the code above?

NbElements per block=16/4=4elements.

3

Between the address of A[i] and the address of A[i+256], there are $256 * 4 = 1024$ bytes, and the low 10 bits of the address will be the same. The two addresses will go in the same line and there will therefore be 1 cache miss on each access, i.e. a 100% miss rate.

ii. Give two software techniques (code modification) and one hardware technique (cache modification) that would achieve the minimum miss rate considering startup faults.

The lines are 16 bytes, or 4 integers. It is therefore necessary to access all the elements of a line before it is ejected;

First software technique.

Loop unwinding of order 4, i.e. the program

```
int A[512], i ;  
for (i = 0; i < 256; i += 4) { A[i] = A[i] + 1;  
  
    A[i+1] = A[i+1] + 1; A[i+2] = A[i+2]  
    + 1; A[i+3] = A[i+3] + 1; A[i+256] =  
    A[i+256] - 1; A[i+257] = A[i+257] -  
    1; A[i+258] = A[i+258] - 1;  
    A[i+259] = A[i+259] - 1; }
```

Second software technique

Break the loop into two separate loops

```
int A[512], i ;  
for (i = 0; i < 256; i++) A[i] = A[i] +  
1;  
for (i = 0; i < 256; i++) A[i+256] =  
A[i+256] - 1;
```

Break the loop into two separate loops

```
int A[512], i ;  
for (i = 0; i < 256; i++) {  
  
    A[i] = A[i] + 1; A[511-i] = A[511-i] - 1;  
    ; }
```

Hardware technology

Use a set-associative cache (two-way).

Hardware technology

Use a set-associative cache (two-way).

iii. So what is the failure rate?

Miss rate = 25% (lines have 4 words. Accessing the first word causes a cache miss).

4

Exercise 4: (35 pts)

Consider the following RISC assembly code on which we wish to perform execution analyses.

1	ADD	R1,	R2,	R3
2	SUB	R5,	R5,	R1
3	L.D.	R4,	(R7)	
4	MUL	R4,	R4,	R5
5	ST	(R7),	R4	
6	DIV	R1,	R7,	R4
7	AND	R5,	(R7),	R1
8	L.D.	R9,	(R12)	
9	ASH	R11,	R11,	R12
10	GOLD	R12,	R9,	R11
11	LSH	R11,	R11,	R9
12	ST	(R11),	R11	
13	MUL	R10	R11,	#2
14	ADD	R14,	R11,	R12

has. Determine the data hazards that appear in this code by specifying which instructions and their type (read after write, write after write and write after read).

LAE: (20)	ADD,SUB,R1	SUB, MUL,R5	LD, DIV, R4
	ADD,AND,R1	LD,MUL,R4	MUL, ST, R4
	DIV,AND,R1	LD, ST, R4	MUL, DIV, R4
	LD, OR, R9	LD, LSH, R9	ASH, GOLD, R11
	ASH, LSH, R11	ASH, ST, R11	ASH, MUL, R11
	ASH, ADD, R11	OR, ADD, R12	LSH, ST, R11
	LSH, MUL, R11	LSH, ADD, R11	
EAE (4)	ADD, D IV, R1	LD, MUL, R4	SUB,AND,R5
	ASH, LSH, R11		
EAL (8)	DIV, SUB, R1	AND, MUL, R5	GOLD, LD, R12
	OR, LSH, R11	AND, SUB, R5	ASH, LSH, R11
	ASH, GOLD, R12		

b. What is the longest chain of dependent operations in the program fragment?

ADD R1,R2,R3-SUB R5,R5,R1-LD R4, (R7)-MUL R4,R4,R5-DIV R1,R7,R4- AND R5, (R7),R1

5

vs. Calculate the execution time of the assembly code knowing that the processor executes the instructions on a 5-stage IO pipeline and that it uses the Bubble solution for conflict resolution (without Bypass). Draw the pipeline to easily count execution cycles.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
EI	ADD	SUB	LD	MUL	MUL	MUL	ST	DIV	DIV	DIV	AND	AND	AND	LD	ASH	ASH	ASH	OR
Di		ADD	SUB	LD	LD	LD	MUL	ST	ST	ST	DIV	DIV	DIV	AND	LD	LD	LD	ASH
LR			ADD	SUB	SUB	SUB	LD	MUL	MUL	MUL	ST	ST	ST	DIV	AND	AND	AND	LD
EX				ADD	Nop	Nop	SUB	LD	Nop	Nop	MUL	Nop	Nop	ST	DIV	Nop	Nop	AND
ER					ADD	Nop	Nop	SUB	LD	Nop	Nop	MUL	Nop	Nop	ST	DIV	Nop	Nop

	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
EI	LSH	ST	ST	MUL	ADD	ADD	ADD											
Di	OR	LSH	LSH	ST	MUL	MUL	MUL	ADD										
LR	ASH	OR	OR	LSH	ST	ST	ST	MUL	ADD									
EX	LD	ASH	Nop	OR	LSH	Nop	Nop	ST	MUL	ADD								
ER	AND	LD	ASH	Nop	OR	LSH	Nop	Nop	ST	MUL	ADD							

d. Rewrite the code by renaming registers and identifying the remaining unresolved issues.

1	ADD	HWR1,	HWR2,	HWR3
2	SUB	HWR25,	HWR5,	HWR1
3	L.D.	HWR4,	(HWR7)	
4	MUL	HWR24,	HWR4,	HWR25
5	ST	(HWR7),	HWR24	
6	DIV	HWR21,	HWR7,	HWR24
7	AND	HWR35,	(HWR7),	HWR21
8	L.D.	HWR9,	(HWR12)	
9	ASH	HWR31,	HWR11,	HWR12
10	GOLD	HWR22,	HWR9,	HWR31
11	LSH	HWR41,	HWR31,	HWR9
12	ST	(HWR41),	HWR41	
13	MUL	HWR10	HWR41,	#2
14	ADD	HWR14,	HWR41,	HWR22

e. Calculate the execution time of the assembly code after renaming knowing that the processor executes them on an OOO superscalar processor equipped with two execution units (one unit is reserved for memory operations only and one for other types

of operations) and whose memory instructions have a latency of two cycles and the other types have a latency of one cycle.

Cycle	Unit 1 (memory operations)	Unit 2 (others)
1	LD HWR4, (HWR7)	ADD HWR1, HWR2, HWR3
2	Busy	SUB HWR5, HWR5, HWR1
3	LD HWR9, (HWR12)	MUL HWR24, HWR4, (HWR7)
4	Busy	DIV HWR21 HWR7, HWR24
5	ST (HWR7), HWR24	AND HWR35, (HWR7), HWR21
6	Busy	ASH HWR31, HWR11, HWR12
7	Free	GOLD HWR22, HWR9, HWR31
8	Free	LSH HWR41, HWR31, HWR9
9	ST (HWR41), HWR41	MUL HWR10, HWR41, #2
10	Busy	ADD HWR14, HWR41, HWR22

f. We assume that we use the Tomasulo algorithm for conflict resolution, with one execution unit for addition/subtraction, and one unit for multiplications/divisions. Each unit has 4 reservation stations. Show the status of the stations and register bank after issuing the original code instructions (all instructions are issued at the same cycle to the stations).

	Op	T1	V1	T2	V2
#1	Add	0	X	0	y
#2	Sub	0	z	1	#1
#3	Add	1	#7	1	#8
#4					

	Op	T1	V1	T2	V2
#5	Div	1	#2	0	t
#6	Mul	1	#5	1	#2
#7	Div	1	#6	0	2
#8	Div	0	t	1	#6

R	B	V
R1	1	#1
	1	#8
R2	0	X
R3	0	y
R4	1	#5
	1	#6
R5	0	z
	1	#2
	1	#3
R6	1	#7
R7	0	t

Exercice 1: (15 pts)

- a. Soit un processeur non pipeliné ayant un chemin de données constitué de modules dont les temps d'exécution sont respectivement : 2, 3, 2, 4, 5, 2, 5, 2, 3 (en cycles). Quel est le temps de cycle du processeur non pipeliné ? Proposer une version pipelinée à 6 étages (non forcement égaux) et calculer le nouveau temps de cycle sachant que le temps de stabilisation entre les étages est de 1 ns.

Temps de cycle non pipeliné = $2+3+2+4+5+2+5+2+3=28$ cycles.

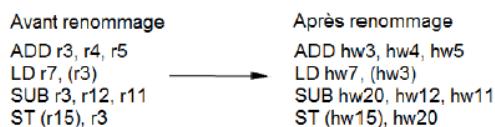
Une version pipelinée possible : 5(2+3), 6(2+4), 5, 2, 5, 5(2+3).

$$\begin{aligned} \text{Temps de cycle pipeliné} &= \text{temps de cycle maximal} + \text{latence} \\ &= 6+1=7 \text{ cycles} \end{aligned}$$

- b. C'est quoi la technique de ré-nommage de registres ? Quels types de conflits peuvent être résolus en utilisant cette technique et lesquels non ? Justifier.

Le renommage des registres est une technique qui vise à minimiser l'impact des dépendances EAL et EAE sur le parallélisme d'instructions en assignant dynamiquement chaque valeur produite par un programme à un nouveau registre, afin de rompre les dépendances EAL et EAE.

La figure suivante illustre la manière dont le renommage de registres peut améliorer la performance.



Exercice 2: (25 pts)

Soit une mémoire cache de taille 32K octets avec une taille de ligne de 16 octets, une associativité d'ordre 8, géré en écriture différée et utilisant une politique de remplacement de type LRU. La mémoire centrale est adressable par octet avec des adresses de 32 bits.

- a. Calculer le nombre de lignes et d'ensembles dans le cache.

$$\text{Nombre de lignes} = \frac{32 \cdot 2^{10}}{16} = \frac{2^{15}}{2^4} = 2^{11} = 2048 \text{ lignes}$$

$$\text{Nombre d'ensembles} = \frac{2^{11}}{2^3} = 2^8 = 256 \text{ lignes}$$

- b. Calculer le nombre de bits de l'étiquette.

$$\text{Largeur de bus d'ad} = 32 = \text{nbBits.} \text{etiquette} + \text{nBits.} \text{ensemble} + \text{nbBits.} \text{Octet}$$

$$32 = \text{nbBits.} \text{etiquette} + \log_2 256 + \log_2 16$$

$$32 = \text{nbBits.} \text{etiquette} + 8 + 4$$

$$32 = \text{nbBits.} \text{etiquette} + 12$$

$$\text{nbBits.} \text{etiquette} = 32 - 12 = 20 \text{ Bits}$$

- c. Pour les adresses mémoires hexadécimales suivantes, donner le numéro d'ensemble, le numéro d'octet référencé dans la ligne et les bits d'étiquettes : 0xAB1DCEFF, 0xEFA00AC0. Donner par la suite un exemple de deux adresses mémoire qui seront mappées sur un même ensemble du cache.

0xAB1DCEFF							
A	B	1	D	C	E	F	F
Etiquette					Ensemble		Octet
1010	1011	0001	1101	1100	1110	1111	1111
AB1DC					EF		F
AB1DC					239		15

0xEFA00AC0							
E	F	A	0	0	A	C	0
Etiquette					Ensemble		Octet
1110	1111	1010	0000	0000	1010	1100	0000
EFA00					AC		0
EFA00					172		0

Deux adresses mémoires mappées sur le même ensemble (ensemble 0) :

Adresse memoire	Etiquette	Ensemble	Octet
0x012AB000	012AB	00	0
0x8CFD100F	8CFD1	00	F

- d. Donner la taille d'une entrée d'étiquette et celle du tableau d'étiquettes.

$$T. l. \text{etiquette} = 1\text{bit}(Valid) + 1(\text{ecr. différé}) + \text{nbBits.} \text{etique} + \text{nbBits.} \text{algo LRU}$$

$$T. l. \text{etiquette} = 1 + 1 + 20 + \log_2 8$$

$$T. l. \text{etiquette} = 1 + 1 + 20 + 3 = 25 \text{ bits}$$

$$\text{Taille du tableau d'etiquettes} = 25 * 2^{11} = 50 * 2^{10} \text{ Bits} = 50 \text{ KBits}$$

- e. Sachant qu'on garde la même taille de cache et de ligne de cache, comment faut-il modifier le reste des paramètres du cache (associativité et politique de mise à jour de la mémoire centrale) pour réduire au maximum la taille du tableau d'étiquettes, donner la nouvelle taille du tableau d'étiquettes.

Pour reduire au maximum la taille du tableau d'étiquettes il faut penser à réduire la taille de l'entrée du tableau d'étiquette.

Pour arriver à faire cette réduction, on doit d'abord changer la politique d'écriture (utiliser l'écriture immédiate) et réduire au maximum le nombre de lignes par ensemble c-à-dire une ligne par ensemble.

Lorsqu'on réduit le nombre de lignes par ensemble on aura une modification au niveau de taille de l'étiquette qu'on doit recalculer.

$$\text{Largeur de bus d'ad} = 32 = \text{nbBits.} \text{étiquette} + \text{nBits.} \text{index} + \text{nbBits.} \text{Octet}$$

$$32 = \text{nbBits.} \text{étiquette} + \log_2 \text{nbLignes} + \log_2 16$$

$$32 = \text{nbBits.} \text{étiquette} + \log_2 2048 + 4$$

$$32 = \text{nbBits.} \text{étiquette} + 11 + 4$$

$$\text{nbBits.} \text{étiquette} = 32 - 15 = 17 \text{ Bits}$$

Donc :

$$T. \text{l'} \text{étiquette} = 1 \text{bit(Validité)} + \text{nbBits.} \text{étiquette} + \text{nbBits.} \text{algo LRU}$$

$$T. \text{l'} \text{étiquette} = 1 + 17 + \log_2 1$$

$$T. \text{l'} \text{étiquette} = 18 \text{ Bits}$$

$$\text{Taille du tableau d'} \text{étiquettes} = 18 * 2^{11} = 36 * 2^{10} \text{ Bits} = 36 \text{ KBits}$$

Exercice 3: (25 pts)

Soit un cache de 1 Ko à correspondance directe, avec des lignes (blocs) de 16 octets. Le processeur a des registres de 32 bits et des adresses de 32 bits. Le cache est à réécriture (write back) et allocation d'écriture (en cas d'échec en écriture, on amène le bloc en mémoire cache pour l'écrire).

- a. Quels sont les nombres de bits nécessaires pour l'index, l'étiquette et le déplacement (adresse dans la ligne) ?

$$1 \text{ Ko} = 2^{10}.$$

$$\text{Lignes de 16 octets : } 2^4.$$

$$\text{Nombre de lignes : } 2^6 = 64.$$

$$\text{Déplacement : 4 bits}$$

$$\text{Index : 6 bits}$$

$$\text{Etiquette : 22 bits}$$

- b. Soit le code C suivant :

```
int A[512], i ;
for (i = 0; i < 256; i++)
{
    A[i] = A[i] + 1;
    A[i+256] = A[i+256] - 1;
}
```

- i. Quel est le taux d'échec (nombre de défauts / nombre d'accès) lors de l'exécution du code ci-dessus ?

NbElements par bloc=16/4=4elements.

Entre l'adresse de A[i] et l'adresse de A[i+256], il y a $256 * 4 = 1024$ octets, et les 10 bits de poids faible de l'adresse seront identiques. Les deux adresses iront dans la même ligne et il y aura donc 1 défaut de cache à chaque accès, soit un taux d'échec de 100%.

- ii. Donner deux techniques logicielles (modification du code) et une technique matérielle (modification du cache) qui permettraient d'obtenir le taux d'échec minimal compte tenu des défauts de démarrage.

Les lignes sont de 16 octets, soit 4 entiers. Il faut donc accéder à tous les éléments d'une ligne avant qu'elle soit éjectée ;

Première technique logicielle.

Déroulage de boucle d'ordre 4, soit le programme

```
int A[512], i ;
for (i = 0; i < 256; i += 4) {
    A[i] = A[i] + 1;
    A[i+1] = A[i+1] + 1;
    A[i+2] = A[i+2] + 1;
    A[i+3] = A[i+3] + 1;
    A[i+256] = A[i+256] - 1;
    A[i+257] = A[i+257] - 1;
    A[i+258] = A[i+258] - 1;
    A[i+259] = A[i+259] - 1;
}
```

Deuxième technique logicielle

Décomposer la boucle en deux boucles distinctes

```
int A[512], i ;
for (i = 0; i < 256; i++)
    A[i] = A[i] + 1;
for (i = 0; i < 256; i++)
    A[i+256] = A[i+256] - 1;
```

Décomposer la boucle en deux boucles distinctes

```
int A[512], i ;
for (i = 0; i < 256; i++)
{
    A[i] = A[i] + 1;
    A[511-i] = A[511-i] - 1;
}
```

Technique matérielle

Utiliser un cache associatif par ensemble (deux voies).

- iii. Quel est alors le taux d'échec ?

Taux d'échec = 25% (les lignes ont 4 mots. L'accès au premier mot provoque un défaut de cache).

Exercice 4 : (35 pts)

Soit le code assembleur RISC suivant sur lequel on souhaite faire des analyses d'exécution.

1	ADD	R1,	R2,	R3
2	SUB	R5,	R5,	R1
3	LD	R4,	(R7)	
4	MUL	R4,	R4,	R5
5	ST	(R7),	R4	
6	DIV	R1,	R7,	R4
7	AND	R5,	(R7),	R1
8	LD	R9,	(R12)	
9	ASH	R11,	R11,	R12
10	OR	R12,	R9,	R11
11	LSH	R11,	R11,	R9
12	ST	(R11),	R11	
13	MUL	R10	R11,	#2
14	ADD	R14,	R11,	R12

- a. Déterminer les aléas de données qui figurent dans ce code en précisant entre quelles instructions et leur type (lecture après écriture, écriture après écriture et écriture après lecture).

LAE : (20)	ADD,SUB,R1	SUB, MUL,R5	LD, DIV , R4
	ADD,AND, R1	LD,MUL,R4	MUL, ST, R4
	DIV, AND, R1	LD, ST, R4	MUL, DIV, R4
	LD, OR, R9	LD, LSH, R9	ASH, OR, R11
	ASH, LSH, R11	ASH, ST, R11	ASH, MUL, R11
	ASH, ADD, R11	OR, ADD, R12	LSH, ST, R11
	LSH, MUL, R11	LSH, ADD, R11	
EAE (4)	ADD, D IV, R1	LD, MUL, R4	SUB, AND, R5
	ASH, LSH, R11		
EAL (8)	DIV, SUB, R1	AND, MUL, R5	OR, LD, R12
	OR, LSH, R11	AND, SUB, R5	ASH, LSH, R11
	ASH, OR, R12		

- b. Quelle est la plus longue chaîne d'opérations dépendantes du fragment de programme ?

ADD R1,R2,R3 → SUB R5,R5,R1 → LD R4, (R7) → MUL R4,R4,R5 → DIV R1,R7,R4 → AND R5,(R7),R1

- c. Calculer le temps d'exécution du code assembleur sachant que le processeur exécute les instructions sur un pipeline IO de 5 étages et qu'il utilise la solution des Bulles pour la résolution des conflits (sans Bypass). Dessiner le pipeline pour compter facilement les cycles d'exécutions.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
EI	ADD	SUB	LD	MUL	MUL	MUL	ST	DIV	DIV	DIV	AND	AND	AND	LD	ASH	ASH	ASH	OR
Di		ADD	SUB	LD	LD	LD	MUL	ST	ST	ST	DIV	DIV	DIV	AND	LD	LD	LD	ASH
LR			ADD	SUB	SUB	SUB	LD	MUL	MUL	MUL	ST	ST	ST	DIV	AND	AND	AND	LD
EX				ADD	Nop	Nop	SUB	LD	Nop	Nop	MUL	Nop	Nop	ST	DIV	Nop	Nop	AND
ER					ADD	Nop	Nop	SUB	LD	Nop	Nop	MUL	Nop	Nop	ST	DIV	Nop	Nop

	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
EI	LSH	ST	ST	MUL	ADD	ADD	ADD											
Di	OR	LSH	LSH	ST	MUL	MUL	MUL	ADD										
LR	ASH	OR	OR	LSH	ST	ST	ST	MUL	ADD									
EX	LD	ASH	Nop	OR	LSH	Nop	Nop	ST	MUL	ADD								
ER	AND	LD	ASH	Nop	OR	LSH	Nop	Nop	ST	MUL	ADD							

- d. Réécrire le code en faisant un ré-nommage de registres et identifier les aléas restant non résolues.

1	ADD	HWR1,	HWR2,	HWRR3
2	SUB	HWR25,	HWR5,	HWR1
3	LD	HWR4,	(HWR7)	
4	MUL	HWR24,	HWR4,	HWR25
5	ST	(HWR7),	HWR24	
6	DIV	HWR21,	HWR7,	HWR24
7	AND	HWR35,	(HWR7),	HWR21
8	LD	HWR9,	(HWR12)	
9	ASH	HWR31,	HWR11,	HWR12
10	OR	HWR22,	HWR9,	HWR31
11	LSH	HWR41,	HWR31,	HWR9
12	ST	(HWR41),	HWR41	
13	MUL	HWR10	HWR41,	#2
14	ADD	HWR14,	HWR41,	HWR22

- e. Calculer le temps d'exécution du code assembleur après ré-nommage sachant que le processeur exécute les sur un processeur superscalaire OOO doté de deux unités d'exécution (une unité est réservée pour les opérations mémoire seulement et une pour les autres types

d'opérations) et dont les instructions mémoires possèdent une latence de deux cycles et les autres types possèdent une latence d'un cycle.

Cycle	Unité 1 (opérations mémoires)	Unité 2 (autres)
1	LD HWR4, (HWR7)	ADD HWR1, HWR2, HWR3
2	Occupé	SUB HWR5, HWR5, HWR1
3	LD HWR9, (HWR12)	MUL HWR24, HWR4, (HWR7)
4	Occupé	DIV HWR21 HWR7, HWR24
5	ST (HWR7), HWR24	AND HWR35, (HWR7), HWR21
6	Occupé	ASH HWR31, HWR11, HWR12
7	Libre	OR HWR22, HWR9, HWR31
8	Libre	LSH HWR41, HWR31, HWR9
9	ST (HWR41), HWR41	MUL HWR10, HWR41, #2
10	Occupé	ADD HWR14, HWR41, HWR22

- f. On suppose qu'on utilise l'algorithme de Tomasulo pour la résolution des conflits, avec une unité d'exécution pour les addition/soustraction, et une unité pour les multiplications/divisions. Chaque unité dispose de 4 stations de réservation. Montrer l'état des stations et du banc de registres après l'émission des instructions du code original (toutes les instructions sont émises au même cycle vers les stations).

	Op	T1	V1	T2	V2
#1	Add	0	X	0	y
#2	Sub	0	z	1	#1
#3	Add	1	#7	1	#8
#4					

	Op	T1	V1	T2	V2
#5	Div	1	#2	0	t
#6	Mul	1	#5	1	#2
#7	Div	1	#6	0	2
#8	Div	0	t	1	#6

R	B	V
R1	1	#1
	1	#8
R2	0	X
R3	0	y
R4	1	#5
	1	#6
R5	0	z
	1	#2
	1	#3
R6	1	#7
R7	0	t

Bonne chance



COMPUTER ARCHITECTURE I 2207 – ENGLISH

Exercise 1 (50 points)

Write a MIPS program that:

- a- Declares two arrays A and B of 20 (32-bit) integers each one;
- b- Asks the user to fill the array A with 20 positive integers (your program does not accept negative or null values);
- c- Stores the array A in reverse order in the array B;

.data

A: .space 80
B: .space 80

Message1:.ascii "enter a positive integer: "

.text

```
    li      $t0, 0
    li      $t1, 0
    li      $t2, 0
    li      $t3, 20
    la      $a1, A
    la      $a2, B
Loop:   li      $v0, 4
        la      $a0, Message1
        syscall
        li      $v0, 5
        syscall
        blez   $v0, Loop
        addi   $t1, 1
        sw      $v0, 0($a1)
        addi   $a1, $a1, 4
        bne    $t1, $t3, Loop
Loop2:  li      $t1, 0
        la      $a1, A
        lw      $t2, 76($a1)
        sw      $t2, 0($a2)
        addi   $a2, $a2, 4
        addi   $a1, $a1, -4
        addi   $t1, 1
        bne    $t1, $t3, Loop2
        li      $v0, 10
syscall
```



COMPUTER ARCHITECTURE I 2207 – ENGLISH

Exercise 1 (50 points)

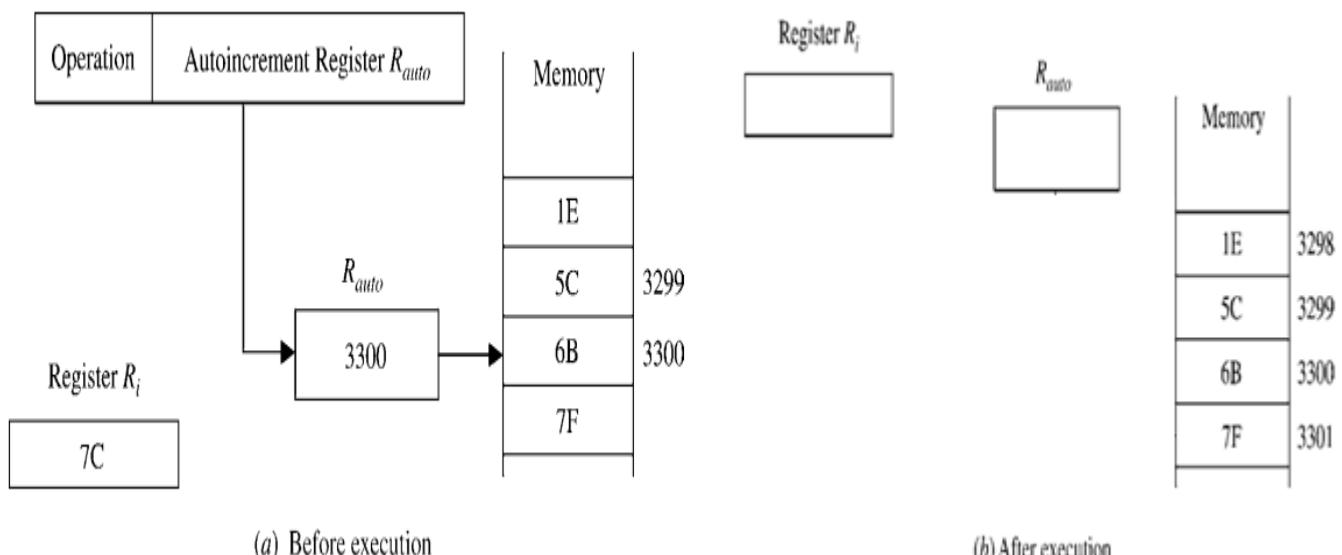
Write a MIPS program that:

- Declares two arrays A and B of 20 (32-bit) integers each one;
- Asks the user to fill the array A with 20 positive integers (your program does not accept negative or null values);
- Stores the array A in reverse order in the array B;

Exercise 2 (20 points)

Complete, by filling the new registers values, the following pictures for each of the following operations:

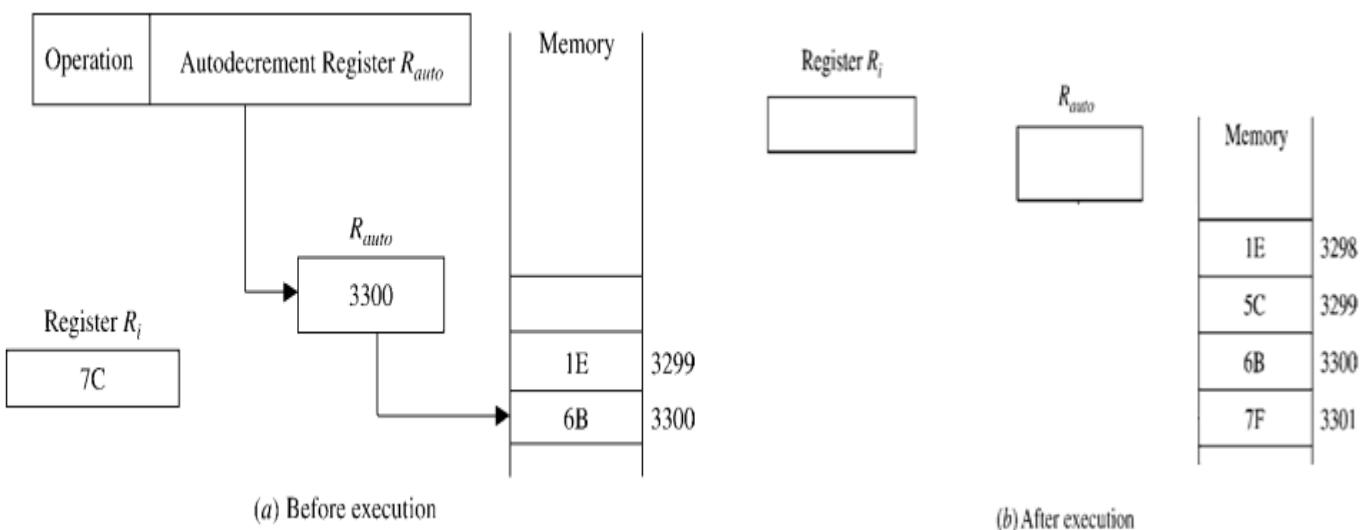
a- LOAD (R_{auto})+, R_i



(a) Before execution

(b) After execution

b- LOAD -(R_{auto}), R_i



(a) Before execution

(b) After execution

Exercise 3 (30 points)

Consider the instruction **Add X, R0** that adds the contents of memory location X to register R0 and stores the result in R0.

Give the steps (and micro-actions) performed by the cpu to execute this instruction in each of this case:

- *One-bus datapath;*
- *Two-bus datapath;*
- *Two-bus datapath (in – out buses)*
- *Three-bus datapath*

Some MIPS instructions and directives you may use (you may use others)

Arithmetic Instructions

Instruction	Example	Meaning	Comments
add	add \$1,\$2,\$3	\$1=\$2+\$3	
add immediate	addi \$1,\$2,100	\$1=\$2+100	"Immediate" means a constant number
Divide	div \$2,\$3	\$Hi:\$low=\$2/\$3	Remainder stored in special register hi Quotient stored in special register lo

Assembler Directives

.ascii str	Store the ASCII string str in memory. Strings are in double-quotes, i.e. "Computer Science"
.asciiz str	Store the ASCII string str in memory and null-terminate it Strings are in double-quotes, i.e. "Computer Science"
.space n	Leave an empty n-byte region of memory for later use
.align n	Align the next datum on a 2^n byte boundary. For example, .align 2 aligns the next value on a word boundary

System Calls

Service	Operation	Code (in \$v0)	Arguments
print_int	Print integer number (32 bit)	1	\$a0 = integer to be printed
print_float	Print floating-point number (32 bit)	2	\$f12 = float to be printed
print_double	Print floating-point number (64 bit)	3	\$f12 = double to be printed
print_string	Print null-terminated character string	4	\$a0 = address of string in memory
read_int	Read integer number from user	5	None

Conditional Branch

Instruction	Example	Meaning
branch on equal	beq \$1,\$2,100	if(\$1==\$2) go to PC+4+100
branch on not equal	bne \$1,\$2,100	if(\$1!=\$2) go to PC+4+100
branch on greater than	bgt \$1,\$2,100	if(\$1>\$2) go to PC+4+100
branch on greater than or equal	bge \$1,\$2,100	if(\$1>=\$2) go to PC+4+100
branch on less than	blt \$1,\$2,100	if(\$1<\$2) go to PC+4+100
branch on less than or equal	ble \$1,\$2,100	if(\$1<=\$2) go to PC+4+100

Data Transfer

Instruction	Example	Meaning
load word	lw \$1,100(\$2)	\$1=Memory[\$2+100]
store word	sw \$1,100(\$2)	Memory[\$2+100]=\$1
load address	la \$1,label	\$1=Address of label
load immediate	li \$1,100	\$1=100

Good luck



COMPUTER ARCHITECTURE I 2207 – ENGLISH

Exercise 1 (50 points)

Write a MIPS program that:

- a- Declare an array of 200 (32-bit) integers;
- b- Asks the user to fill the array with 200 positive integers (your program does not accept negative or null values);
- c- Calculates and prints on the screen the summation of the even elements in the array;

```
.data
Array:    .space 800
Message1: .asciiz "enter a positive integer"
Message2: .asciiz "the sum of the even elements is"
.text
li      $t0, 0
li      $t1, 0
li      $t3, 200
li      $t2, 2
la      $a1, Array
Loop:   li      $v0, 4
        la      $a0, Message1
        syscall
        li      $v0, 5
        syscall
        blez   $v0, Loop
        addi   $t1, 1
        sw     $v0, 4($a1)
        div    $v0, $t2
        mfhi   $t4
        bnez   $t4, Test
        add    $t0, $t0, $v0
Test:   bne   $t1, $t3, Loop
        li      $v0, 4
        la      $a0, Message2
        syscall
        li      $v0, 1
        mov   $a0, $t0
        syscall
        li      $v0, 10
        syscall
```

Exercise 2 (25 points)

Consider the instruction **Add X, R0** that adds the contents of memory location X to register R0 and stores the result in R0.

Give the steps (and micro-actions) performed by the CPU to execute this instruction in each of these cases:

A. *One-bus organization;*

MAR \leftarrow X
 MDR \leftarrow Mem[MAR]
 $A \leftarrow (R0); B \leftarrow (MDR)$
 $R0 \leftarrow (A) + (B)$

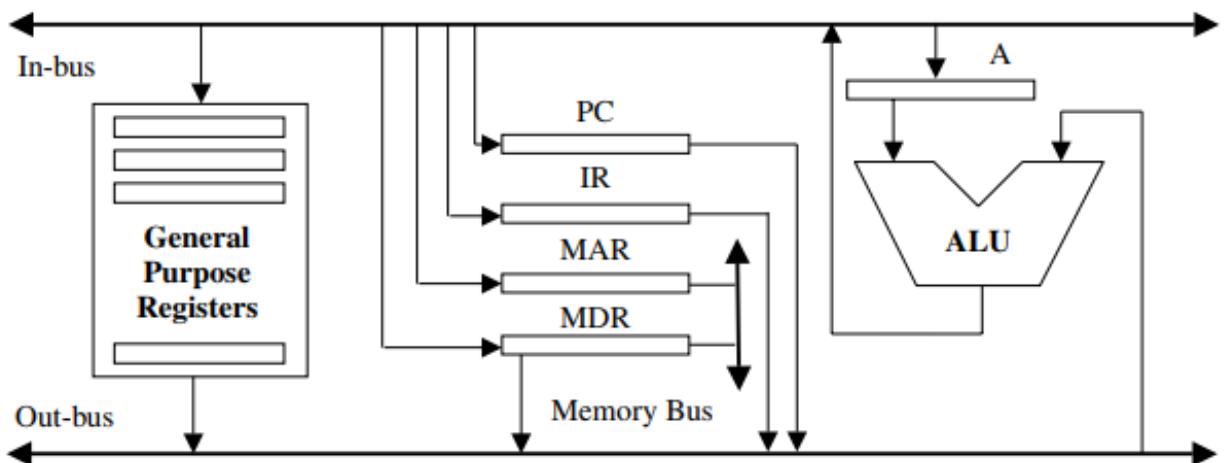
B. *Three-bus organization.*

MAR \leftarrow X
 MDR \leftarrow Mem[MAR]
 $R0 \leftarrow (R0) + (MDR)$

Exercise 3 (25 points)

Design a logic circuit that corresponds to the control signal generated by the control unit to add the contents of the registers R1 and R2 and to store the result in the register R0 using the following two-bus (in – out buses) datapath

Step	type	micro-operation			
t0	Inst-x	$A \leftarrow (R1)$	R1 to in-bus	in-bus to A	
t1	Inst-x	$R0 \leftarrow (A) + (R2)$	R2 to out-bus	ALU add	in-bus to R0



Good luck

Some MIPS instructions and directives you may use (you may use others)

Arithmetic Instructions

Instruction	Example	Meaning	Comments
add	add \$1,\$2,\$3	\$1=\$2+\$3	
add immediate	addi \$1,\$2,100	\$1=\$2+100	"Immediate" means a constant number
Divide	div \$2,\$3	\$hi,\$low=\$2/\$3	Remainder stored in special register \$hi Quotient stored in special register \$lo

Conditional Branch

Instruction	Example	Meaning
branch on equal	beq \$1,\$2,100	if(\$1==\$2) go to PC+4+100
branch on not equal	bne \$1,\$2,100	if(\$1!=\$2) go to PC+4+100
branch on greater than	bgt \$1,\$2,100	if(\$1>\$2) go to PC+4+100
branch on greater than or equal	bge \$1,\$2,100	if(\$1>=\$2) go to PC+4+100
branch on less than	blt \$1,\$2,100	if(\$1<\$2) go to PC+4+100
branch on less than or equal	ble \$1,\$2,100	if(\$1<=\$2) go to PC+4+100

System Calls

Service	Operation	Code (in \$v0)	Arguments
print_int	Print integer number (32 bit)	1	\$a0 = integer to be printed
print_float	Print floating-point number (32 bit)	2	\$f12 = float to be printed
print_double	Print floating-point number (64 bit)	3	\$f12 = double to be printed
print_string	Print null-terminated character string	4	\$a0 = address of string in memory
read_int	Read integer number from user	5	None

Assembler Directives

.ascii str	Store the ASCII string str in memory. Strings are in double-quotes, i.e. "Computer Science"
.asciiz str	Store the ASCII string str in memory and null-terminate it Strings are in double-quotes, i.e. "Computer Science"
.space n	Leave an empty n-byte region of memory for later use
.align n	Align the next datum on a 2^n byte boundary. For example, .align 2 aligns the next value on a word boundary

Data Transfer

Instruction	Example	Meaning
load word	lw \$1,100(\$2)	\$1=Memory[\$2+100]
store word	sw \$1,100(\$2)	Memory[\$2+100]=\$1
load address	la \$1,label	\$1=Address of label
load immediate	li \$1,100	\$1=100



Cours : Info 2207

Durée : 1^h 30^m

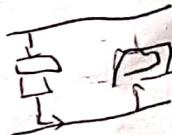
Année : 2020-2021

Examen : 2^{ème}

Exercice 1: (30 pts)

Considérer l'instruction suivante **MUL R0, X, R0** qui permet de multiplier le contenu de l'adresse mémoire X au contenu du registre R0 et enregistre le résultat dans le registre R0.

1. Donner les étapes (et les micro-actions) [cycle d'instruction] effectuées par le CPU pour fetch and execute de cette instruction dans chacun de cas suivant :
 - Two-bus organization;
 - Three-bus organization.
2. Quels sont les avantages et les inconvénients de chaque organisation ?



Exercice 2: (35 points)

Étant donné l'organisation de la mémoire. La mémoire centrale est composée 128 KMots. La mémoire cache est composée de 128 blocs de 32 mots d'un octet. Le cache est à réécriture différée (write back).

1. Comment l'unité de gestion de mémoire interprète une adresse d'un mot de la mémoire principale sachant que le cache est à correspondance directe ?
2. Calculer la taille de l'étiquette ?
3. Calculer la taille totale de la mémoire cache ?
4. En considérant le schéma suivant (l'état du cache), expliquer toutes les étapes qui permettent de satisfaire la requête faite par le processeur pour accéder à l'élément qui se trouve à l'adresse 00010000011000100

Tag(Etiquette)	
0	1
1	3
2	4
3	2
4	1
5	3
6	2
7	31
127	6

Cache	
0	Block 128
1	Block 1
2	Block 130
3	Block 515
4	Block 516
5	Block 389
6	Block 134
7	Block 7
127	Block 255

A ← R₀ . Q₀

Page 1 of 2

0 1

Exercice 3: (35 pts)

Le processeur dispose d'une mémoire cache associative de 3 entrées. Les adresses mémoire sont sur 16 bits. Chaque mot mémoire fait 64 bits. La mémoire centrale est adressable par octet. Chaque entrée du cache contient un bloc de 4 mots.

- Quelle est la taille de la mémoire centrale?
- Quelle est la taille de l'étiquette?
- Soit la suite de références suivantes, qui correspondent aux accès mémoire demandés par le processeur, en terme d'adresses d'octets, dans le temps. Les adresses sont données en hexadécimal (base 16)

Temps	0	1	2	3	4	5	6	7	8	9	10
Adresse	002F	AABB	1148	002D	ABCD	0016	012B	ABD2	1137	013C	1B33

Donner l'évolution des 3 entrées du répertoire du cache et noter les défauts et les succès ainsi que le pourcentage d'échec dans les deux cas suivants:

- la politique de remplacement est FIFO.
- la politique de remplacement est LRU.

00000

0000 0010.

0000 0000 0010 1111

Bonne chance



COMPUTER ARCHITECTURE I 2207 – ENGLISH

Exercise 1 (40 points)

Write a MIPS program that:

- a- Declares an array A of 10 (32-bit) integers;
- b- Asks the user to fill the array A with 10 positive integers (your program does not accept negative or null values);
- c- Prints the maximum value and the number of times it occurs in the array;

Example: if A is

1	4	-2	13	4	12	5	13	3	-2
---	---	----	----	---	----	---	----	---	----

The program displays: *the maximum is 13 it occurs 2 time(s)*

```
.data
Array:    .space 40
Message1: .asciiz "enter an integer "
Message2: .asciiz "the max is "
Message3: .asciiz " it appears "
Message4: .asciiz " time(s). "
.text
# t1 is the number of inserted elements
li      $t1, 1
# t2 is the max
li      $t2, 0
# t3 is the number of elements to insert
li      $t3, 10
# t4 is the number of occurrences of the max
li      $t4, 1
la      $a1, Array
li      $v0, 4
la      $a0, Message1
syscall
li      $v0, 5
syscall
sw      $v0, 0($a1)
mov   $t2, $v0
Loop: li      $v0, 4
la      $a0, Message1
syscall
li      $v0, 5
syscall
Eq: addi   $t4, 1
Test: bne   $t1, $t3, Loop
li      $v0, 4
la      $a0, Message2
syscall
li      $v0, 1
mov   $a0, $t2
syscall
li      $v0, 4
la      $a0, Message3
syscall
li      $v0, 1
mov   $a0, $t4
syscall
la      $a0, Message4
syscall
li      $v0, 10
syscall
```

Exercise 2 (30 points)

Consider a program's instruction **Add R0, X, R0** that adds the contents of memory location X to register R0 and stores the result in R0.

1. Give the steps (and micro-actions) [instruction cycle] performed by the CPU to **fetch and execute** this instruction in each of this case:

– *One-bus organization;*

Fetching (6 pts)

1- MAR \leftarrow (PC); A \leftarrow (PC)

2- MDR \leftarrow Mem[MAR]; PC \leftarrow (A) + 1

3- IR \leftarrow (MDR)

Executing (6 pts)

MAR \leftarrow X;
MDR \leftarrow Mem[MAR];
A \leftarrow (MDR); B \leftarrow (R0);
R0 \leftarrow (A) + (B);

– *Three-bus organization.*

Fetching (6 pts)

1- MAR \leftarrow (PC); PC \leftarrow (PC) + 1

2- MDR \leftarrow Mem[MAR]

3- IR \leftarrow (MDR)

Executing (6 pts)

MAR \leftarrow X;
MDR \leftarrow Mem[MAR];
R0 \leftarrow (R0) + (MDR)

2. What are the advantages and the disadvantages of each organization?

One-Bus: (3 pts)

Advantages:

- simplest,
- Least expensive

Disadvantage:

- limits the amount of data transfer ... (many actions to execute an instruction)

Three-Bus: (3 pts)

Advantage:

- Fastest organization

Disadvantage:

- Expensive
- Increase the complexity of the hardware

Exercise 3 (30 points)

Given the following memory organization:

	Main Memory					
	0	128	256	384		3968
Cache	0	384	129	257	385	
	1					
	2					
126						
127	4095		127	255	383	4095
	0	1	2	3		31

Where the main memory consists of 4096 (4k) blocks, the cache memory consists of 128 blocks and the block size is 32 words.

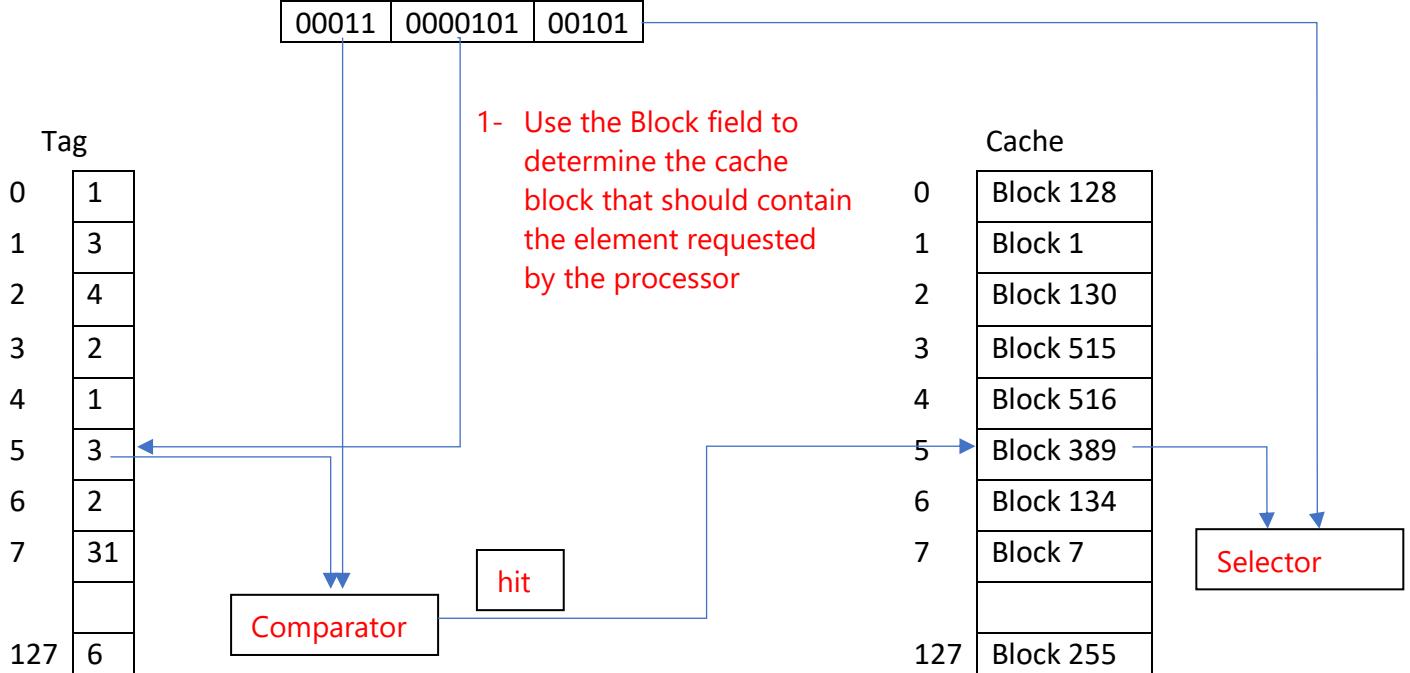
- How the MMU interprets a word's main memory address according to direct-mapping technique? (3*3=9 pts)

Word field: $\log_2 32 = 5$

Block field: $\log_2 128 = 7$

Tag field: $\log_2 (4096/128) = 5$

- Considering the following schema, describe the protocol used by the MMU to satisfy a request made by the processor for accessing the element in the address: (3*7=21 pts)



- Check the corresponding Tag memory to see whether there is a match between its content and that of the Tag field.

- The targeted word can be selected using the Word field.

Exercice 1: (30 pts)

Considérer l'instruction suivante **Add R0, X, R0** qui permet d'ajouter le contenu de l'adresse mémoire X au contenu du registre R0 et enregistre le résultat dans le registre R0.

1. Donner les étapes (et les micro-actions) [cycle d'instruction] effectuées par le CPU pour **fetch and execute** de cette instruction dans chacun de cas suivant :
 - *One-bus organization;*
 - *Three-bus organization.*

One-bus organization :

Fetching :

- 1- MAR \leftarrow (PC); A \leftarrow (PC)
- 2- MDR \leftarrow Mem[MAR]; PC \leftarrow (A) + 1
- 3- IR \leftarrow (MDR)

Executing :

- MAR \leftarrow X;
 MDR \leftarrow Mem[MAR];
 A \leftarrow (MDR); B \leftarrow (R0);
 R0 \leftarrow (A) + (B);

Three-bus organization:

Fetching :

- 1- MAR \leftarrow (PC); PC \leftarrow (PC) + 1
- 2- MDR \leftarrow Mem[MAR]
- 3- IR \leftarrow (MDR)

Executing :

- MAR \leftarrow X;
 MDR \leftarrow Mem[MAR];
 R0 \leftarrow (R0) + (MDR)

2. Quels sont les avantages et les inconvénients de chaque organisation ?

One-bus organization:

Advantages:

- simplest,
- Least expensive

Inconvenients::

- limits the amount of data transfer ... (many actions to execute an instruction)

Three-bus organization:

Advantage:

- Fastest organization

Disadvantage:

- Expensive
- Increase the complexity of the hardware

Exercice 2: (30 points)

Étant donné l'organisation de la mémoire. La mémoire centrale est composée 128 KMots. La mémoire cache est composée de 128 blocs de 32 mots d'un octet. Le cache est à réécriture différée (write back).

1. Comment l'unité de gestion de mémoire interprète une adresse d'un mot de la mémoire principale sachant que le cache est à correspondance directe ?
Voir cours (Fonctionnement de la moire cache a correspondance directe)
2. Calculer la taille de l'étiquette ?

Taille du mot=1 octet

Taille de la mémoire centrale= $128 \text{ KMots} = 128 \times 2^{10} \text{ Mots} = 2^7 \times 2^{10} \text{ octets} = 2^{17} \text{ octets}$

Largeur de bus d'adresse= $\log_2 \text{taille de la mmoire centrale} = \log_2 2^{17} = 17 \text{ bits}$

La mémoire est à correspondance directe, donc l'adresse est de la forme suivante :

Adresse (17bits)		
Etiquette	index	Num. octet

Largeur de bus adresse = Nb. bre de bits par étiquette+ Nb bits index+Nb de bits par octet

Nb bits par index= $\log_2 \text{nb de lignes} = \log_2 128 = 7 \text{ bits}$ (nb blocs=nb de lignes)

Nb bits par octet= $\log_2 \text{nb octet par ligne} = \log_2 32 = \log_2 2^5 = 5 \text{ bits}$

Nombre de bits par étiquette= 17- nombre de bits par octet-nb bits par index

Nombre de bits par étiquette= 17-5-7=5 bits

3. Calculer la taille totale de la mémoire cache ?

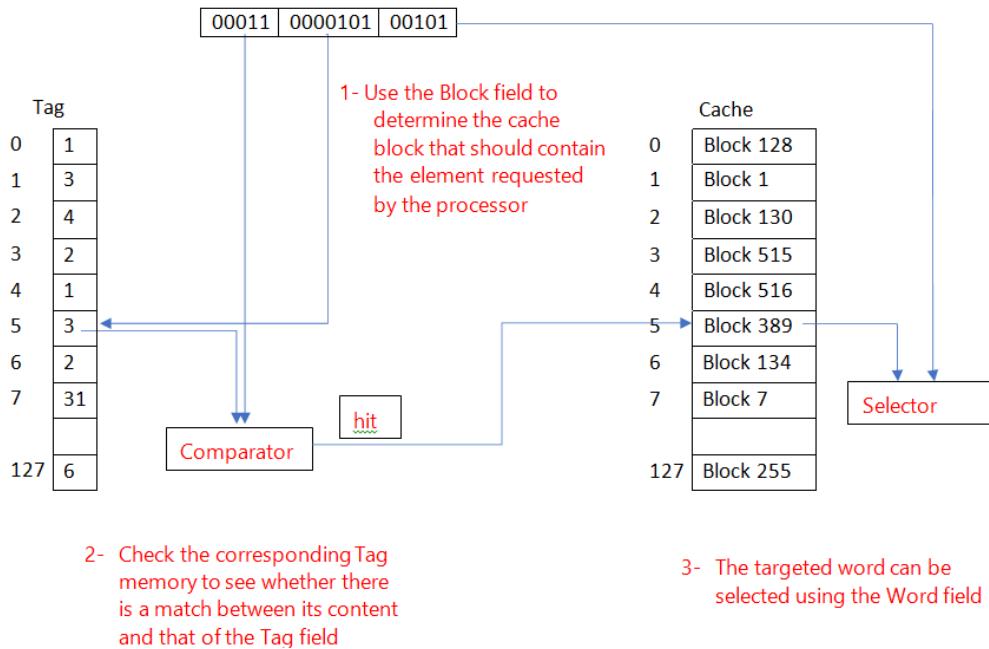
Taille d'une ligne du cache=1(bit de validité)+ 1(bit de modification) +5 (bits d'étiquette)+32*8(nombre de bits dans une ligne données utiles)

Taille d'une ligne=7+256=263 bits

Taille totale du cache=nb de lignes* taille de la ligne=128*263 bits

4. En considérant le schéma suivant (l'état du cache), expliquer toutes les étapes qui permettant de satisfaire la requête faite par le processeur pour accéder à l'élément qui se trouve à l'adresse 00011000010100101

Tag(Etiquette)		Cache
0	1	0 Block 128
1	3	1 Block 1
2	4	2 Block 130
3	2	3 Block 515
4	1	4 Block 516
5	3	5 Block 389
6	2	6 Block 134
7	31	7 Block 7
127	6	127 Block 255



Exercice 3: (40 pts)

Le processeur dispose d'une mémoire cache associative de 4 entrées. Les adresses mémoire sont sur 16 bits. Chaque mot mémoire fait 32 bits. La mémoire centrale est adressable par octet. Chaque entrée du cache contient un bloc de 4 mots.

- a. Quelle est la taille de la mémoire centrale?

Largeur de bus adresse= 16 bits

La mémoire est adressable par octet, donc la taille de la mémoire centrale est :

$$2^{16} \text{ octets} = \frac{2^{16}}{2^{10}} K\text{Octets} = 2^6 K\text{Octets} = 64 K\text{Octets}$$

- b. Quelle est la taille de l'étiquette?

Nombre d'entrees (nombre des lignes)=4

Largeur de bus adresse= 16 bits

Taille d'un mot= 32 bits=4 octets.

Nombre d'octets par entree=4 mots=4*4octets= 16 octets

La mémoire est associative, donc l'adresse est de la forme suivante :

Adresse (16bits)	
Etiquette	Num. octet

Largeur de bus adresse=16 bits= nombre de bits par étiquette+ nombre de bits par octet

Nombre de bits par octet= \log_2 nombre d'octets par entrée = $\log_2 16 = 4$ bits

Nombre de bits par étiquette= 16- nombre de bits par octet=16-4=12 bits

- c. Soit la suite de références suivantes, qui correspondent aux accès mémoire demandés par le processeur, en terme d'adresses d'octets, dans le temps. Les adresses sont données en hexadécimal (base 16)

Temps	0	1	2	3	4	5	6	7	8	9	10
Adresse	002F	AABB	1148	004D	ABCD	0016	012B	1B32	1137	003C	1B33

Donner l'évolution des 4 entrées du répertoire du cache et noter les défauts dans les deux cas suivants:

- la politique de remplacement est FIFO.
- la politique de remplacement est LRU.

Temps	0	1	2	3	4	5	6	7	8	9	10
Adresse	002F	AABB	1148	004D	ABCD	0016	012B	1B32	1137	003C	1B33
Etiquette Sur 12 bits	002	AAB	114	004	ABC	001	012	1B3	113	003	1B3

L'étiquette correspondante à chaque adresse est calculée de la manière suivante :

Prenons la première adresse 002F :

L'adresse en Hexadécimale :	002F	
L'adresse en binaire	0000 0000 0010 1111	
	Etiquette (12bits)	Noctet(4 bits)
Adresse =Etiquette Noctet	0000 0000 0010	1111
En Hexadécimale	002	F

Donc, il est évident que les trois valeurs de l'adresse en Hexadécimale représentent l'étiquette.

FIFO

Adresse	002F	AABB	1148	004D	ABCD	0016	012B	1B32	1137	003C	1B33
Etiquette	002	AAB	114	004	ABC	001	012	1B3	113	003	1B3
L1	<u>002</u>	002	002	002	<u>ABC</u>	ABC	ABC	ABC	<u>113</u>	113	113
L2		<u>AAB</u>	AAB	AAB	AAB	<u>001</u>	001	001	001	<u>003</u>	003
L3			<u>114</u>	114	114	114	<u>012</u>	012	012	012	012
L4				<u>004</u>	004	004	004	<u>1B3</u>	1B3	1B3	<u>1B3</u>
	D	D	D	D	D	D	D	D	D	D	S

LRU

Adresse	002F	AABB	1148	004D	ABCD	0016	012B	1B32	1137	003C	1B33
Etiquette	002	AAB	114	004	ABC	001	012	1B3	113	003	1B3
L1	<u>002</u>	002	002	002	<u>ABC</u>	ABC	ABC	ABC	<u>113</u>	113	113
L2		<u>AAB</u>	AAB	AAB	AAB	<u>001</u>	001	001	001	<u>003</u>	003
L3			<u>114</u>	114	114	114	<u>012</u>	012	012	012	012
L4				<u>004</u>	004	004	004	<u>1B3</u>	1B3	1B3	<u>1B3</u>
	D	D	D	D	D	D	D	D	D	D	S

Bonne chance



Course: I2207

Year: 2021-2022

Duration: 1h30m

Review: 1timeSession

Exercise 1: (40 pts)

Given the organization of memory. The central memory is made up of 256 KWords. The cache memory is made up of 64 blocks of 32 one-byte words. The cache is write back.

- How does the memory management unit interpret a one-word address from main memory knowing that the cache is direct matching?

The memory management unit receives an address of a memory word. It interprets the address by detecting the corresponding index (i) and label (e) and the requested byte (o). It will check if line i of the label table contains label (e).

- If this is the case (we speak of success), that is to say the requested word is found in the cache memory. Then it will point to the byte (o) in the payload index line (i) in the cache. And this byte will be delivered to the processor.
- Otherwise (we speak of failure or fault), that is to say the requested word does not exist in the cache. Then, it will put the label (e) in the label table at index (i) and will replace the line (i) of user data from the cache memory with a new line which will be imported from the memory central. Then, the address byte (o) of this line will be sent to the processor.

- Calculate the size of the label?

Word size=1 byte Main
memory size=Address bus
width=

The memory is direct correspondence, so the address is of the following form:

Address (18bits)		
Label	index	Num. byte

Address bus width = No. number of bits per label + number of index bits + number of bits per byte number

of bits per index = (number of blocks=number of lines)

Number of bits per byte=

Number of bits per label= 18- number of bits per byte-number of bits per index
Number of bits per label= 18-5-6=7 bits

- Calculate total cache size?

Size of a cache line = 1 (validity bit) + 1 (modification bit) +7 (label bits) + 32*8
(number of bits in a useful data line)

Line size=9+256=265 bits

Total cache size=number of lines* line size=64*265 bits= 8*265 Bytes

4. Considering the following diagram (the state of the cache), explain all the steps which make it possible to satisfy the request made by the processor to access the element located at the address 101000110110011100

We calculate the index and the label then we explain how the processor will interact with it. N byte: $(11100)_2 = (28)_{10}$

Index: $(101100)_2 = (44)_{10}$ - this is line 28

Label: $(1010001)_2 = (81)_{10}$ - this is label 81

Entry 28 (of the etiauette table) is empty. So the data we are looking for does not exist in the cache. So the processor goes to the central memory to retrieve the data located at the address $(1010001101100\ 00000)_2$ (we can calculate the exact address and entry 28 of the label table will be replaced by the label: (1010001)

Tag		Hidden	
0	5	0	Block 64
1	2	1	Block 65
2	4	2	Block 2
3	7	3	Block 131
4	2	4	Block 196
5	3	5	Block 197
6	2	6	Block 6
7	31	7	Block 7
128	1010001		Block X
31	11	31	Block 255

Exercise 2: (60 pts)

Consider a cache memory with a capacity of 256 KB (kilobyte), with blocks (cache lines) of 128 bytes and a degree of associativity of 8 (set = 8 blocks). The cache is managed in "writeback" mode with an LRU (Least Recently Used) replacement policy. A memory address is coded over 32 bits.

has. How many lines and sets does the cache have?

b. How many entries are required in the tag table (the TAG directory) **The number of entries required in the label table is equal to 2048 labels.**

vs. How many tag (TAG) bits are required for each entry in the tag table?

The cache is mixed of order 8. So we have:

No. of bits/label+No. of bits/set+No. of bits/byte=address bus width.

Then: Number of bits / label + 8 + 7 = 32 - Number of bits / label = 32-15 = 17 bits

d. For the following hexadecimal memory addresses, give the set number, the byte number referenced in the line and the label bits: 0xAB1DCEFF, 0xEFAD0AC0. Subsequently give an example of two memory addresses which will be mapped onto the same cache set.

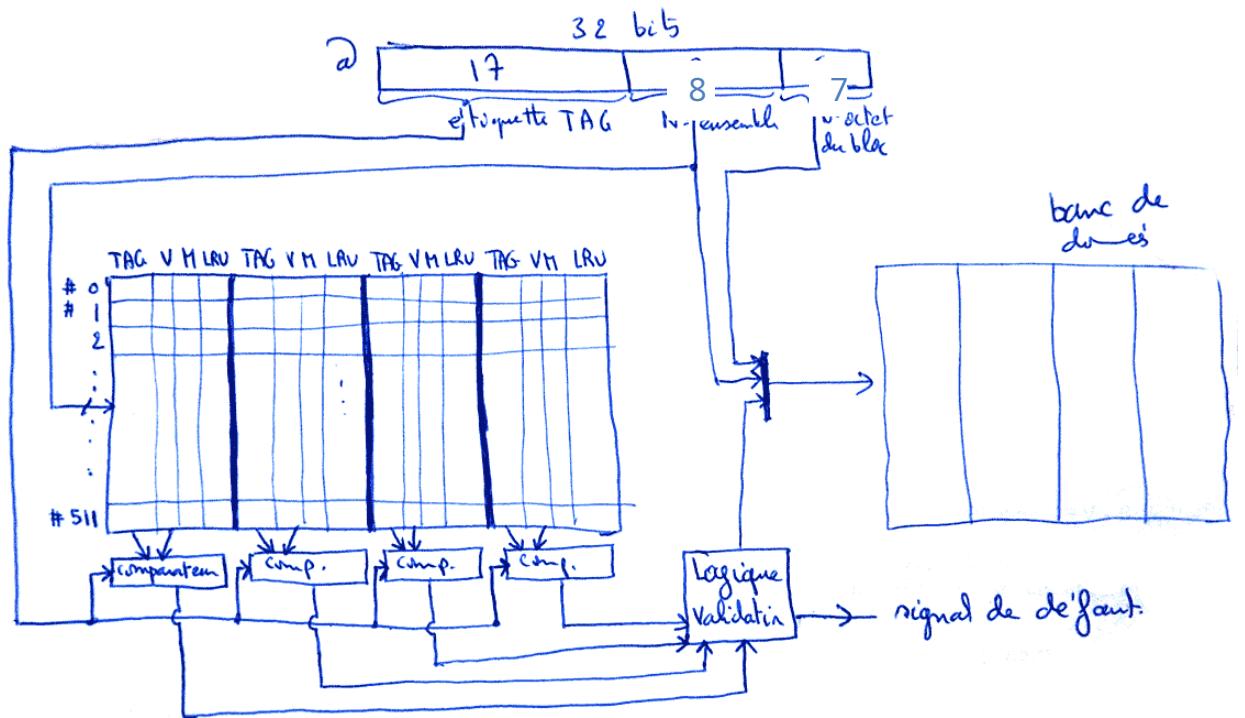
- AB1DCEFF=1010 1011 0001 1101 1100 1110 1111 1111
Label (on 17 bits): 1010 1011 0001 1101 1=(1 563B)_H
Set: 100 1110 1=(9D)_H=(...)₁₀
Byte: 1111111=(127)₁₀=(7F)_H
- EFAD0AC0=1110 1111 1010 1101 0000 1010 1100,0000
Label (on 17 bits): 1110 1111 1010 1101 0=(1DF5A)_H
Set: 000 1010 1=(21)₁₀=(15)_H
Byte: 1000000=(64)₁₀=(40)_H

e. How many bits are required for each integer entry in the label array and how much total memory is required for the array?

No. of bits for each set = 4 x (17 + 1[V] + 1[M] + 3[LRU]) = 88 bits

Table size = 2048 x 88 bits = 256x88 bytes =256*4*22 Bytes=22 KBytes.

f. Draw the complete diagram of the cache memory showing the decomposition of the



Exercise 3: Practical exam (25 pts)

(الإجابة على ذات الكراس)

1. Recall the role of the PSW status register of a processor and the meanings of the bits: z, c, o, s, p.

z: to indicate if the result of the last arithmetic operation is **null**. c: to indicate if the result of the last operation generates a deduction (**Carry**). o: to indicate whether the result of the last arithmetic operation generates **Overflow**. s: to indicate whether the result of the last arithmetic operation is **negative**. p: to indicate whether the result of the last arithmetic operation is **positive**.

2. What is the result of the following operations performed on an 8-bit processor that uses two's complement representation for negative integers? (10 pts, 2.5 each)

has. LSH -14.3

14 in 8-bit binary: 0000 1110

- 14 in C1 on 8 bits : 11110001

- 14 in C2 on 8 bits : 11110010

LSH-14.3 - 10010000 in C2 therefore: -112

10010000 in C2 - 10001111 in C1-01110000 binary-112

b. ASH -17.5

17 in 8-bit binary: 00010001

- 17 in C1 on 8 bits : 11101110

- 17 in C2 on 8 bits : 11101111

ASH-17.5 -11100000 in C2 therefore: -32
 11100000 in C2-11011111 in C1-00100000 binary-32
 vs. LSH 32, -2
 32 in binary on 8 bits: 0010 0000 32
 in C2 on 8 bits: 0010 0000 LSH 32,
 -2 -0000 1000 in binary-8
 d. ASH 32, 2
 32 in binary on 8 bits: 0010 0000 32
 in C2 on 8 bits: 0010 0000 ASH 32, 2
 -1000 0000 in binary--1

3. Write in assembler a program for stack architecture which allows you to calculate the expression:
 $((((-10 8) (4 7 -) +) 4 +) ((-10 8) (4 7 -) +) 4 +) 6 -)$

First, we must write the expression in postfix form:

$((((-10 8) (4 7 -) +) 4 +) ((-10 8) (4 7 -) +) 4 +) 6 -)$

Push -10	Mul
Push 8	Push 4
Mul	Push 7
Push 4	Sub
Push 7	Add
Sub	Push 4
Add	Add
Push 4	Mul
Add	Push 6
Push -10	Sub
Push 8	

Good luck

Exercice 1: (40 pts)

Étant donné l'organisation de la mémoire. La mémoire centrale est composée 256 KMots. La mémoire cache est composée de 64 blocs de 32 mots d'un octet. Le cache est à réécriture différée (write back).

1. Comment l'unité de gestion de mémoire interprète une adresse d'un mot de la mémoire principale sachant que le cache est à correspondance directe ?

L'unité de gestion de mémoire reçoit une adresse d'un mot mémoire. Il interprète l'adresse en détectant l'index (i) et l'étiquette (e) correspondant et l'octet demandé (o). Il va vérifier si la ligne i du tableau d'étiquettes contient l'étiquette (e).

- Si c'est le cas (on parle de succès), c'est -à-dire le mot demandé se trouve dans la mémoire cache. Alors, il va pointer sur l'octet (o) dans la ligne d'index (i) de données utiles dans la mémoire cache. Et cet octet va être libre au processeur.
- Sinon, (on parle d'échec ou défaut), c'est-à-dire le mot demandé n'existe pas dans le cache. Alors, il va mettre l'étiquette (e) dans le tableau d'étiquette à l'index (i) et va remplacer la ligne (i) de données utiles de la mémoire cache par une nouvelle ligne qui va être importée de la mémoire centrale. Puis, l'octet d'adresse (o) de cette ligne va être envoyé au processeur.

2. Calculer la taille de l'étiquette ?

Taille du mot=1 octet

Taille de la mémoire centrale= $256 \text{ KMots} = 256 \times 2^{10} \text{ Mots} = 2^8 \times 2^{10} \text{ octets} = 2^{18} \text{ octets}$

Largeur de bus d'adresse= $\log_2 \text{taille de la mémoire centrale} = \log_2 2^{18} = 18 \text{ bits}$

La mémoire est à correspondance directe, donc l'adresse est de la forme suivante :

Adresse (18bits)		
Etiquette	index	Num. octet

Largeur de bus adresse = Nb. de bits par étiquette + Nb bits index + Nb de bits par octet

Nb bits par index= $\log_2 \text{nb de lignes} = \log_2 64 = 6 \text{ bits}$ (nb blocs=nb de lignes)

Nb bits par octet= $\log_2 \text{nb octet par ligne} = \log_2 32 = \log_2 2^5 = 5 \text{ bits}$

Nombre de bits par étiquette= 18 - nombre de bits par octet - nb bits par index

Nombre de bits par étiquette= 18-5-6=7 bits

3. Calculer la taille totale de la mémoire cache ?

Taille d'une ligne du cache=1(bit de validité)+1(bit de modification) +7 (bits d'étiquette)+32*8(nombre de bits dans une ligne données utiles)

Taille d'une ligne=9+256=265 bits

Taille totale du cache=nb de lignes* taille de la ligne=64*265 bits=8*265 Octets

4. En considérant le schéma suivant (l'état du cache), expliquer toutes les étapes qui permettant de satisfaire la requête faite par le processeur pour accéder à l'élément qui se trouve à l'adresse **101000110110011100**

On calcule l'index et l'étiquette puis on explique comment le processeur va interagir avec.

N octet : $(11100)_2 = (28)_{10}$

Index : $(101100)_2 = (44)_{10} \rightarrow$ c'est la ligne 28

Etiquette : $(1010001)_2 = (81)_{10} \rightarrow$ c'est l'étiquette 81

L'entrée 28 (du tableau d'étiquette) est vide. Donc les données qu'on cherche n'existe pas dans le cache. Donc le processeur va à la mémoire centrale pour récupérer les données qui se trouvent à l'adresse $(1010001101100\ 00000)_2$ (on peut calculer l'adresse exacte X) et mettre les données importer dans l'entrée 28 du tableau de données utiles du cache. et l'entrée 28 du tableau d'étiquette sera remplacée par l'étiquette : (1010001)

Tag(Etiquette)		Cache
0	5	0 Block 64
1	2	1 Block 65
2	4	2 Block 2
3	7	3 Block 131
4	2	4 Block 196
5	3	5 Block 197
6	2	6 Block 6
7	31	7 Block 7
128	1010001	Block X
31	11	31 Block 255

Exercice 2: (60 pts)

Soit une mémoire cache de capacité 256 Ko (kilo octet), avec des blocs (lignes de cache) de 128 octets et un degré d'associativité de 8 (ensemble = 8 blocs). Le cache est géré en mode « write-back » avec une politique de remplacement LRU (Least Recently Used). Une adresse mémoire est codée sur 32 bits.

- a. Combien de lignes et d'ensembles possèdent le cache ?

$$\text{Nombre de lignes} = \frac{\text{taille du cache}}{\text{taille de ligne}} = \frac{256 * 2^{10}}{128} = \frac{2^{18}}{2^7} = 2^{11} = 2048 \text{ lignes}$$

$$\text{Nombre d'ensembles} = \frac{\text{Nombre de lignes}}{\text{Nombre de lignes par ensemble}} = \frac{2048}{8} = \frac{2^{11}}{2^3} = 2^8 = 256 \text{ ensembles}$$

- b. Combien d'entrées sont requises dans le tableau d'étiquettes (le répertoire des TAG)

Le nombre d'entrées requises dans le tableau d'étiquettes est égal à 2048 étiquettes.

- c. Combien des bits d'étiquette (TAG) sont requis pour chaque entrée dans le tableau d'étiquettes ?

Le cache est mixte d'ordre 8. Donc nous avons :

Nb de bits/etiquette+Nb de bits/ensemble+Nombre de bits/octet=largeur de bus d'adresse.

Nombre de bits_ensemble = \log_2 nombre d'ensembles = $\log_2 256 = \log_2 2^8 = 8$ bits

Nombre de bits_octet = \log_2 nombre d'octets par ligne = $\log_2 128 = \log_2 2^7 = 7$ bits

Alors : Nombre de bits /etiquette+8+7=32 → Nombre de bits /etiquette=32-15=17 bits

- d. Pour les adresses mémoires hexadécimales suivantes, donner le numéro d'ensemble, le numéro d'octet référencé dans la ligne et les bits d'étiquettes : 0xAB1DCEFF, 0xEFAD0AC0. Donner par la suite un exemple de deux adresses mémoire qui seront mappées sur un même ensemble du cache.

- AB1DCEFF=1010 1011 0001 1101 1100 1110 1111 1111

Etiquette (sur 17 bits) : 1010 1011 0001 1101 1=(1563B)_H

Ensemble : 100 1110 1=(9D)_H =(...)₁₀

Octet : 1111111=(127)₁₀=(7F)_H

- EFAD0AC0=1110 1111 1010 1101 0000 1010 1100 0000

Etiquette (sur 17 bits) : 1110 1111 1010 1101 0=(1DF5A)_H

Ensemble : 000 1010 1=(21)₁₀=(15)_H

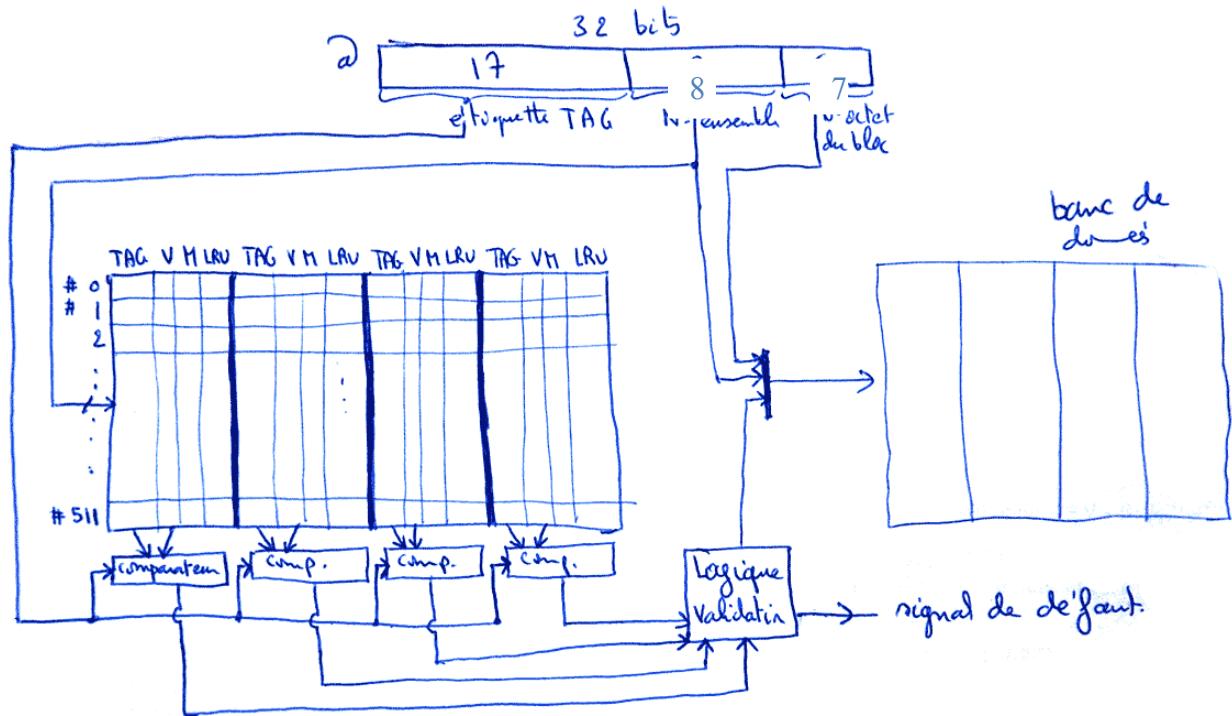
Octet : 1000000=(64)₁₀=(40)_H

- e. Combien des bits sont requis pour chaque entrée entière du tableau d'étiquettes et quelle quantité de mémoire totale est requise pour le tableau ?

Nb. de bits pour chaque ensemble = $4 \times (17 + 1[V] + 1[M] + 3[LRU]) = 88$ bits

Taille du tableau = 2048×88 bits = 256×88 octets = $256 \times 4 \times 22$ Octets=22 KOctets.

- f. Dessiner le schéma complet de la mémoire cache en montrant la décomposition du



Exercice 3: Examen TP(25 pts)

(الإجابة على ذات الكراس)

- Rappeler le rôle du registre d'état PSW d'un processeur et les significations des bits : z, c, o, s, p.
 - z : pour indiquer si le résultat de dernière opération arithmétique est **nul**.
 - c : pour indiquer si le résultat de dernière opération génère un retenu (**Carry**).
 - o : pour indiquer si le résultat de dernière opération arithmétique génère **Overflow**.
 - s : pour indiquer si le résultat de dernière opération arithmétique est **négatif**.
 - p : pour indiquer si le résultat de dernière opération arithmétique est **positif**.
- Quel est le résultat des opérations suivantes exécutées sur un processeur 8 bits qui utilise la représentation en complément à deux pour les entiers négatifs ? (10 pts, 2.5 chacune)
 - LSH -14,3
 - 14 en binaire sur 8 bits : 0000 1110
 - 14 en C1 sur 8 bits : 11110001
 - 14 en C2 sur 8 bits : 11110010
 - LSH-14,3 → 10010000 en C2 donc : -112
 - 10010000 en C2 → 10001111 en C1 → 01110000 binaire → 112
 - ASH -17,5
 - 17 en binaire sur 8 bits : 00010001
 - 17 en C1 sur 8 bits : 11101110
 - 17 en C2 sur 8 bits : 11101111

ASH-17,5 → 11100000 en C2 donc : -32
 11100000 en C2 → 11011111 en C1 → 00100000 binaire → 32
 c. LSH 32, -2
 32 en binaire sur 8 bits : 0010 0000
 32 en C2 sur 8 bits : 0010 0000
 LSH 32, -2 → 0000 1000 en binaire → 8
 d. ASH 32, 2
 32 en binaire sur 8 bits : 0010 0000
 32 en C2 sur 8 bits : 0010 0000
 ASH 32, 2 → 1000 0000 en binaire → -1

3. Ecrivez en assembleur un programme pour architecture à pile qui permet de calculer l'expression : $((-10 \times 8) + (4 - 7) + 4)^2 - 6$

D'abord, on doit écrire l'expression sous forme postfixée :

$((((-10 \times 8) (4 7 -) +) 4 +) (((-10 8 \times) (4 7 -) +) 4 +) \times) 6 -)$

Push -10	Mul
Push 8	Push 4
Mul	Push 7
Push 4	Sub
Push 7	Add
Sub	Push 4
Add	Add
Push 4	Mul
Add	Push 6
Push -10	Sub
Push 8	

Bonne chance



Exercise 1- (35 points) Write a MIPS program that prompts the user to enter two positive integers, x and y, and calculates the quotient of x divided by y without using the division operator. This can be made by repeating subtract y from x until x becomes less than y.

Exercice 1- (35 points) Écrire un programme MIPS qui demande à l'utilisateur de saisir deux entiers positifs x et y, puis calcule le quotient de la division de x par y sans utiliser l'opérateur de division. Cela peut être réalisé en soustrayant y de x de manière répétée jusqu'à ce que x devienne inférieur à y.

```
# Variable declarations
.data
prompt1:    .asciiz "Enter the first integer x:"
prompt2:    .asciiz "Enter the second integer y:"
result:     .asciiz "x/y="

.text
.globl main

main:
# Prompt the user to enter x
li $v0, 4
la $a0, prompt1
syscall
li $v0, 5
syscall
move $t0, $v0

# Prompt the user to enter y
li $v0, 4
la $a0, prompt2
syscall
li $v0, 5
syscall
move $t1, $v0
# Initialize the quotient to zero
li $t2, 0

loop:
# Check if x is less than y
slt $t3, $t1, $t0
beqz $t3, exit

# Subtract y from x
```

```

sub $t0, $t0, $t1

# Increment the quotient
addi $t2, $t2, 1

j loop

exit:
# Display the quotient
li $v0, 4
la $a0, result
syscall
li $v0, 1
move $a0, $t2
syscall

# Terminate the program
li $v0, 10
syscall

```

Exercise 2- (25 points)

- 1- Represent the CPU micro-operations corresponding to fetching an instruction from memory in a:
 - a- one- bus organization.
 - b- Three-buses organization.
- 2- Represent the CPU micro-operations corresponding to executing an instruction from the IR in a:
 - c- one- bus organization.
 - d- Three-buses organization.

Exercice 2 - (25 points)

- 1- Représenter les micro-opérations du CPU correspondant à la récupération (fetching) d'une instruction depuis la mémoire dans le cas de :
 - a- Une organisation à un bus.
 - b- Une organisation à trois bus.
- 2- Représenter les micro-opérations du CPU correspondant à l'exécution d'une instruction depuis le registre IR dans :
 - c- Une organisation à un bus.
 - d- Une organisation à trois bus.

Fetching 1-bus :

1- $\text{MAR} \leftarrow (\text{PC})$; $\text{A} \leftarrow (\text{PC})$

2- $\text{MDR} \leftarrow \text{Mem}[\text{MAR}]$; $\text{PC} \leftarrow (\text{A}) + 1$

3- $\text{IR} \leftarrow (\text{MDR})$

Fetching 3-bus :

1- $\text{MAR} \leftarrow (\text{PC})$; $\text{PC} \leftarrow (\text{PC}) + 1$

2- $\text{MDR} \leftarrow \text{Mem}[\text{MAR}]$

3- $\text{IR} \leftarrow (\text{MDR})$

Instruction : ADD R1,R2,R0

Executing 1-bus:

$\text{A} \leftarrow (\text{R1})$

$\text{B} \leftarrow (\text{R2})$

$\text{R0} \leftarrow (\text{A}) + (\text{B})$

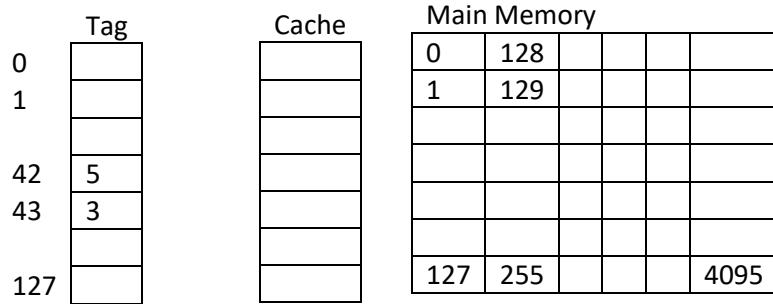
Executing 3-bus :

$\text{R0} \leftarrow (\text{R1}) + (\text{R2})$

Exercise 3- (**40 points**) Consider a computer system with a direct mapping technique and the following case (figure) of a main memory consisting of 4K blocks, a cache memory consisting of 128 blocks, and a block size of 16 words.

- a- In which cache the bloc 129 of the main memory must be placed before being used by the CPU?
- b- According to the direct-mapping technique, the MMU interprets the address issued by the processor by dividing the address into three fields, give the size and the name of each field.
- c- How the word address 0X1ABC is interpreted?

- d- Is the main memory bloc containing the word OX1ABC present in the cache memory?
Justify your answer.



Exercice 3- (40 points) Considérer un système informatique utilisant une technique de mappage direct et les paramètres suivants (figure) : une mémoire principale composée de blocs de 4K, une mémoire cache composée de 128 blocs et une taille de bloc de 16 mots.

- a- Dans quelle cache le bloc 129 de la mémoire principale doit-il être placé avant d'être utilisé par le CPU ?
- b- Selon la technique de mappage direct, l'unité de gestion de la mémoire (MMU) interprète l'adresse émise par le processeur en divisant l'adresse en trois champs. Donnez la taille et le nom de chaque champ.
- c- Comment l'adresse du mot 0X1ABC est-elle interprétée ?
- d- Le bloc de la mémoire principale contenant le mot 0X1ABC est-il présent dans la mémoire cache ? Justifiez votre réponse.

a- $129 \bmod 128 = 1$, so the memory bloc 129 must be present in the second cache which number is 1

b- Word field which size is $\log_2 B$, where B is the size of the block in words= 4
Block field which size is $\log_2 N$, where N is the size of the cache in blocks= 7

Tag field which size is $\log_2 (M/N)$, where M is the size of the main memory in blocks= 5

c- 0X1ABC=

0001 1010 1011 1100

00011 0101011 1100

- Calculating the content of the Bloc field= $1+2+8+32=43$

- Comparing the tag field value with the tag number 43. The value of the tag field = $1+2=3$

The value on the tag table 43 = 3 = the value of the tag field
so it hits