Fiche de TD N° 5 Architecture des Ordinateurs (AO)

Exercice 1

Soit l'expression suivante : **A = E – (F / G)**

1- Réécrire cette expression en une suite d'instructions de **format à deux (02) adresses**.
2- Réécrire cette expression en une suite d'instructions de **format à une (01) adresse.**
3- Réécrire cette expression en une suite d'instructions de **format à zéro (0) adresse.**

Exercice 2

Trouvez les résultats du fragment de programme suivant pour les trois modes d'adressage que voici : **Immédiat, Direct et Indirect.**

ADD 10
SUB 20
MPY 30
DIV 10
Sachant que [acc]=50 ; [10] =30 ; [20] = 10 ; [30] =20.

Exercice 3

Donnez l'expression de X effectuée par le programme suivant en mode direct
Sachant que [70]=A; [50]=B; [3]=C; [160]=X.

1. LOAD 70
2. ADD 50
3. MPY 3
4. STORE 100
5. LOAD 50
6. DIV 70
7. SUB 3
8. ADD 100
9. STA 160

Exercice 4

Soit l'instruction d'affectation **X=** (A-B)*(C+D/E) telles que A, B, C, D et E sont des variables préalablement définies et stockées respectivement dans les adresses A, B, C, D et E. Ecrire les programmes correspondant aux différents cas suivants :

- Instructions à 2 adresses (avec un seul registre)
- Instructions à 1 adresse
- Instructions à 0 adresse

Exercice 5

a) Décrivez les différentes **étapes d'exécution des instructions** :

9 LOAD 10 ;
10 ADD 25 ;
11 STA 30 ;

b) Dérouler le petit programme suivant, sachant que : [acc]=50 ; [30] =39 ; [31] =12 ;

10 ADD 30
11 DIV 31
12 STA 32
13 Branch si S=0/-4

c) Ecrire un programme en assembleur pour le fragment suivant : i= 0; while (i < 10) i++; avec une machine a 3 adresses.

Exercice 6

1. On considère une machine à **zéro adresse.** Ecrire le programme qui permet de calculer **X** tel que :
   X = (A+B) / (B*(E+C)) en utilisant le minimum d'instructions possible où A, B, C et E sont des adresses. (Donner l'état de la pile pour chaque instruction).
2. Ecrire le programme qui permet de calculer **X** dans une **machine à trois adresses** à l'aide des commandes suivantes :

MPY Adr, Ri, R(i+1);	**ADD Ri, R(i+1), R(i+2);**	**DIV Ri, R(i+1), R(i+2);**
LOAD Ri , im VAL ;	**MOVE Adr, Ri ;**	**STA Ri, Adr ;**

Avec i= 1…n, im= adressage immédiat, VAL = Valeur où Ri, R(i+1), R(i+2) sont des registres internes et Adr est une adresse mémoire.

3. Expliquer l'étape de recherche de l'opérande et d'exécution brièvement (Phase 2 des étapes d'exécution de l'instruction) de l'instruction suivante : **MOVE A, R1.**

Exercice 7

1. Écrire le programme qui permet de calculer **X** dans **une machine à 1 adresse** où A, B, C, D, E, F et G sont des adresses et R1, R2, R3, … sont des registres internes (donner l'état du registre ACC pour chaque instruction).

   X = A - (B / (C+D)) + (E*F) – G

2. **S**achant qu'un **accès mémoire dure 2ns** et qu'**une opération au niveau de l'UAL dure 3ns**. Donnez **la durée totale nécessaire pour exécuter ce programme**. Justifiez votre réponse.
3. Supposons qu'on a **deux ordinateurs A et B** ayant un **temps d'exécution d'une instruction** égal à **5 cycles d'horloge** chacun. **L'ordinateur A** dispose de la technique de traitement **pipeline** et **l'ordinateur B ne l'est pas** (exécution séquentielle). On veut exécuter un programme de 10 instructions. Pouvez-vous en déduire lequel des **ordinateurs A et B est le plus rapide** a exécuté ce programme ?
2. Supposons qu'on a deux ordinateurs. Un **ordinateur A** avec un **processeur cadencé à 3000 MHz** et un **CPI** (nombre de cycles par instruction) de **5 cycles**, et **l'ordinateur B** avec un **processeur cadencé à 2000 MHz** et un **CPI de 4 cycles**. Quel **ordinateur est le plus rapide** et **de combien de fois**.

Fiche de TD N° 5 Architecture des Ordinateurs (AO) (Solution)

Exercice 1

Soit l'expression suivante : **A = E – (F / G)**

<table>
<tr><th colspan="2">Instructions à 2 adresses</th></tr>
<tr><td>(avec un seul registre)
MOVE G, R1 //[R1]=G
DIV F, R1 //[R1]=F/G
SUB E, R1 //[R1]=E-(F/G)
STORE R1, A // [R1]=[A]= E-(F/G)</td><td>(avec des registres)
MOVE G, R1 //[R1]=G
MOVE F, R2 //[R2]=F
DIV R1, R2 //[R2]=F/G
MOVE E, R3 //[R3]=E
SUB R3, R2 //[R2]=E-(F/G)
STORE R2, A // [R2]=[A]= E-(F/G)</td></tr>
<tr><td>Instructions à 1 adresse
(Accumulateur)</td><td>Instructions à 0 adresse
(pile)</td></tr>
<tr><td>LOAD F; //[ACC] = F
DIV G; //[ACC] = F/G
STA X; //[X]=F/G
LOAD E; //[ACC] =E
SUB X; //[ACC] =E-(F/G)
STORE A; OU STA A; //[A]= E-(F/G)</td><td>PUSH E; //pile={E}
PUSH F; //pile={E,F}
PUSH G; //pile={E,F,G}
DIV; //pile={E,F/G}
SUB; //pile={E-F/G}
POP A; //pile={ } et A= E-(F/G)</td></tr>
</table>

Exercice 2

ADD 10
SUB 20
MPY 30
DIV 10

Sachant que [acc]=50 ; [10] =30 ; [20] = 10 ; [30] =20.

Immédiat

ADD 10 => [Acc] ← [Acc]+10=50+10=60
SUB 20 => [Acc] ← [Acc]-20==60-20=40
MPY 30 => [Acc] ← [Acc]*30=40*30=1200
DIV 10 => [Acc] ← [Acc]/ 10 = 1200/10=120

Direct

ADD 10 => [Acc] ← [Acc]+ [10] =50+30=80
SUB 20 => [Acc] ← [Acc]- [20] =80-10=70
MPY 30 => [Acc] ← [Acc]*[30] =70*20=1400
DIV 10 => [Acc] ← [Acc]/ [10] = 1400/30=46.66

Indirect

ADD 10 => [Acc] ← [Acc]+ [[10]] =50+ [30] =50+20=70
SUB 20 => [Acc] ← [Acc]- [[20]] =70- [10] =70-30=40
MPY 30 => [Acc] ← [Acc]*[[30]] =40*[20] =40*10=400
DIV 10 => [Acc] ← [Acc]/ [[10]] = 400/ [30] =400/20=20

Exercice 3

Donnez l'expression de X effectuée par le programme suivant en mode direct
Sachant que [70]=A; [50]=B; [3]=C; [160]=X.

1. LOAD 70 => [Acc] ← [70] = A
2. ADD 50 => [Acc] ← [Acc] + [50] = A+B
3. MPY 3 => [Acc] ← [Acc]* [3] =(A+B) * C
4. STORE 100 => [100] ← [Acc] =(A+B) *C
5. LOAD 50 => [Acc] ← [50] = B
6. DIV 70 => [Acc] ← [Acc]/ [70] =B/A
7. SUB 3 => [Acc] ← [Acc]- [3] = B/A-C
8. ADD 100 => [Acc] ← [Acc]+ [100] =(B/A-C) +((A+B) *C)
9. STA 160 => [160] ← [Acc] =(B/A-C) +((A+B) *C)

Donc **X=(B/A-C) + ((A+B) *C)**

Exercice 4

Soit l'instruction d'affectation **X=** (A-B)*(C+D/E) telles que A, B, C, D et E sont des variables préalablement définies et stockées respectivement dans les adresses A, B, C, D.

Instructions à 2 adresses (avec un seul registre)	Instructions à 1 adresse (Accumulateur)	Instructions à 0 adresse (pile)

MOVE B, R1
SUB A, R1
STORE R1,Y
MOVE E, R2
DIV D, R2
ADD C, R2
MPY Y, R2
STORE R2, X
	LOAD A;
	SUB B;
	STORE Z;
	LOAD D;
	DIV E;
	ADD C;
	MUL Z;
	STORE X;
		PUSH A;
		PUSH B;
		SUB
		PUSH C;
		PUSH D;
		PUSH E;
		DIV;
		ADD;
		MUL;
		POP X;

Exercice 5

a) Décrivez les différentes **étapes d'exécution des instructions (**en mode direct**)** :

9 LOAD 10 ;

Phase 1 : ***Recherche de l'instruction à traiter***

1.1 (CO)→ R@M avec **(CO)=9** donc **R@M=9**

1.2 Commande de lecture.

1.3 (@10)→ RDM avec le contenu de **@9 = LOAD 10** donc **RDM = LOAD 10**

1.4 (RDM) → RI avec **(RDM)= LOAD 10** donc **RI = LOAD 10**

Phase 2 : ***Décodage de l'instruction et recherche de l'opérande***

2.1 Analyse et décodage du code opération. (**Instruction de charment a un seul opérande avec un mode direct**)

1.2 2.2 (ADOP) → R@M avec **(ADOP)=10** donc **R@M=10**

2.3 Commande de lecture.

2.4 (@10) → RDM
2.5 (RDM) → UAL donc la donnée 1 de l'UAL = (@10) ➔ (ACC)= (@10)
Phase 3 : ***Exécution de l'instruction et passer à l'instruction suivante***
3.1 (CO) +1 → CO donc le nouveau contenu du **CO = 10**

10 ADD 25 ;
Phase 1 : ***Recherche de l'instruction à traiter***
1.1 (CO)→ R@M avec **(CO)=10**
1.2 Commande de lecture.
1.3 (@17)→ RDM avec le contenu de **@10 = ADD 25**
1.4 (RDM) → RI avec **(RDM)= ADD 25**
Phase 2 : ***Décodage de l'instruction et recherche de l'opérande***
2.1 Analyse et décodage du code opération. (**Instruction d'addition a un seul opérande avec un mode direct**)
2.2 (ADOP) → R@M avec **(ADOP)=25**
2.3 Commande de lecture.
2.4 (@40) → RDM
2.5 (RDM) → UAL donc la donnée 1 de l'UAL = (@25)
Phase 3 : ***Exécution de l'instruction et passer à l'instruction suivante***
3.1 Exécution de l'opération (Addition). **[Acc] ← [Acc]+[25]**
3.2 Les drapeaux sont positionnés (registre d'état).
3.3 (CO) +1 → CO donc le nouveau contenu du **CO = 11**

11 STA 30 ;
Phase 1 : ***Recherche de l'instruction à traiter***
1.1 (CO)→ R@M avec **(CO)=11**
1.2 Commande de lecture.
1.3 (@18)→ RDM avec le contenu de **@11 = STA 30**
1.4 (RDM) → RI avec **(RDM)=STA 30**
Phase 2 : ***Décodage de l'instruction et recherche de l'opérande***
2.1 Analyse et décodage du code opération. (**Instruction de stockage a un seul opérande avec un mode direct**)
2.2 (ACC) → RDM
2.3 Commande d'écriture.
2.4 La cellule n°30 est sélectionnée, elle est prête à recevoir la donnée
2.5 (RDM) → @30 0
Phase 3 : ***Exécution de l'instruction et passer à l'instruction suivante***
3.1 (CO) +1 → CO donc le nouveau contenu du **CO = 12**

b) Dérouler le petit programme suivant, sachant que : [acc]=50 ; [30] =39 ; [31] =12 ;

10 ADD 30 => [Acc] <- [Acc]+[30]=50+39=89
11 DIV 31 => [Acc] <- [Acc]/[31]=89/12=7.41
12 STA 32 => [32] <- [Acc]=7.41
13 Branch si S=0/-4 ➔ (Branchement vers l'instruction 10 (CO - 4 = 13+1-4) si le contenu de l'accumulateur est POSITIF (S = Signe et 0=positif))

Dans ce cas CO← 10

Donc ➔ Le programme boucle à l'infini.

c) Ecrire un programme en assembleur pour le fragment suivant : i= 0; while (i < 10) i++;

```
STORE i, 0          //i=0
LOAD R1, 10         //R1=10
LOAD R2, i          //R2=i

NI:  ADD R2, R2, 1 ➔ (selon le TP)        // do {R2=R2+1
     Ou ADD 1, R2, R2 ➔ (selon le cours)  // do {R2=R2+1

     SUB R3, R1, R2 ➔ (selon le TP)       // R3=R1-R2 = 10- i
     Ou SUB R1, R2, R3 ➔ (selon le TP)    // R3=R1-R2 = 10- i

     JNZ NI          // } Jump to NI if the result the last operation is Not Zero ➔ while (R3 !=0)
```

Rappel : les différents branchements

Instruction et synonymes	Signification (en anglais)	Condition nécessaire
JZ	**J**ump if **Z**ero flag is set	Indicateur Zéro à 1
JNZ	**J**ump if **N**o **Z**ero flag is set	Indicateur Zéro à 0
JE	**J**ump if both operand were **E**quals	ou quand les deux opérandes A et B sont égaux (A-B = 0)
JNE	**J**ump if both operand were **N**ot **E**quals	ou quand les deux opérandes A et B ne sont pas égaux (A-B != 0)
JS	**J**ump if **S**ign flag is set	Indicateur de signe à 1 (résultat négatif)
JNS	**J**ump if **N**o **S**ign flag is set	Indicateur de signe à 0 (résultat positif)
JO	**J**ump if **O**verflow flag is set	Indicateur de débordement à 1
JNO	**J**ump if **N**o **O**verflow flag is set	Indicateur de débordement à 0
JNAE	**J**ump if **N**ot **A**bove or **E**qual	Indicateur de retenue à 1 (A < B, c'est à dire A-B < 0, retenue)
JAE	**J**ump if **A**bove or **E**qual	Indicateur de retenue à 0 (A >= B, c'est à dire A-B >= 0, pas de retenue)
JNA	ump if **N**ot **A**bove	Indicateur de retenue à 1 ou indicateur Zéro à 1 (A < B ou A = B, c'est à dire A-B <= 0)
JA	**J**ump if **A**bove	Indicateur de retenue à 0 et indicateur Zéro à 0 (A >= B et A != B, c'est à dire A > B, A-B > 0)

Exercice 6

1- On considère une machine à **zéro adresse.** Ecrire le programme qui permet de calculer **X** tel que :
X = (A+B) / (B*(E+C)) en utilisant le minimum d'instructions possible où A, B, C et E sont des adresses. (Donner l'état de la pile pour chaque instruction).

```
PUSH A      //pile={A}
PUSH B      //pile={A, B}
ADD         //pile={A+B}
PUSH B      //pile={A+B, B}
PUSH E      //pile={A+B, B, E}
PUSH C      //pile={A+B, B, E, C}
ADD         //pile={A+B, B, E+C}
MPY         //pile={A+B, B* (E+C)}
DIV         //pile={(A+B/(B* (E+C))}
POP X       //pile={ } et X= (A+B/(B* (E+C))
```

2- Ecrire le programme qui permet de calculer **X** dans une **machine à trois adresses** à l'aide des commandes suivantes :

MPY Adr, Ri, R(i+1);	**ADD Ri, R(i+1), R(i+2);**	**DIV Ri, R(i+1), R(i+2);**
LOAD Ri , im VAL ;	**MOVE Adr, Ri ;**	**STA Ri, Adr ;**

Avec i= 1…n, im= adressage immédiat, VAL = Valeur où Ri, R(i+1), R(i+2) sont des registres internes et Adr est une adresse mémoire.

```
MOVE A, R1
MOVE B, R2
ADD R1, R2, R3
MOVE E, R4
MOVE C, R5
ADD R4, R5, R6
MPY B, R6, R7
DIV R3, R7, R8
STA R8, X
```

3- Expliquer l'étape de recherche de l'opérande et d'exécution brièvement (Phase 2 des étapes d'exécution de l'instruction) de l'instruction suivante : **MOVE A, R1.**

Phase 2 : ***Décodage de l'instruction et recherche de l'opérande***
2.1 Analyse et décodage du code opération.
2.2 (ADOP) ➔ R@M avec **(ADOP)=A**
2.3 Commande de lecture.
2.4 (A)➔ RDM
2.5 (RDM) ➔R1 donc **R1 = (A)**

Exercice 7

1. Écrire le programme qui permet de calculer **X** dans **une machine à 1 adresse** où A, B, C, D, E, F et G sont des adresses et R1, R2, R3, … sont des registres internes (donner l'état du registre ACC pour chaque instruction).

X = A - (B / (C+D)) + (E*F) – G

```
LOAD C      acc = C
ADD D       acc = C+D
STORE R1    R1 = C+D
```

LOAD B	acc = B
DIV R1	acc = B/ (C+D)
STORE R1	R1 = B/ (C+D)
LOAD E	acc = E
MPY F	acc = E*F
STORE R2	R2 = E*F
LOAD A	acc = A
SUB R1	acc = A-(B/ (C+D))
ADD R2	acc = A-(B/ (C+D)) + (E*F)
SUB G	acc = A-(B/ (C+D)) + (E*F) -G
STORE X	X= A-(B/ (C+D)) + (E*F) -G

2. **S**achant qu'un **accès mémoire dure 2ns** et qu'**une opération au niveau de l'UAL dure 3ns**. Donnez **la durée totale nécessaire pour exécuter ce programme**. Justifiez votre réponse.
   Le programme doit calculer : **X = A - (B / (C+D)) + (E*F) – G**
   Accès mémoire ➔ LOAD A, B, C, D, E, F et G, STORE X➔ nombre = 8
   Opérations UAL ➔ ADD, SUB, MPY, DIV ➔ nombre = 6
   Temps d'exécution ce programme = (Nbre d'accès mémoire * 2) + (Nbre opérations * 3)
   = 16 + 18 = **34 ns**
3. Supposons qu'on a **deux ordinateurs A et B** ayant un **temps d'exécution d'une instruction** égal à **5 cycles d'horloge** chacun. **L'ordinateur A** dispose de la technique de traitement **pipeline** et **l'ordinateur B ne l'est pas** (exécution séquentielle). On veut exécuter un programme de 10 instructions. Pouvez-vous en déduire lequel des **ordinateurs A et B est le plus rapide** a exécuté ce programme ?
   n ➔ nombre d'instructions
   k ➔ nombre de cycles d'horloge pour exécuter une instruction
   Exécution programme par A = k + n -1 = 10+5-1 = 14 Cycles
   Exécution programme par B = k * n = 10*5 = 50 Cycles
   A est plus rapide que B pour exécuter ce programme
3. Supposons qu'on a deux ordinateurs. Un **ordinateur A** avec un **processeur cadencé à 3000 MHz** et un **CPI** (nombre de cycles par instruction) de **5 cycles**, et **l'ordinateur B** avec un **processeur cadencé à 2000 MHz** et un **CPI de 4 cycles**. Quel **ordinateur est le plus rapide** et **de combien de fois**.
   MIPS (A) = fréquence / CPI = 3000 / 5 = 600
   MIPS (B) = fréquence / CPI = 2000 / 4 = 500
   A est plus rapide que B
   Combien de fois = MIPS (A)/ MIPS (B) = 600/500= 1.2