DIRO
IFT 1214

EXAMEN FINAL

Max Mignotte

DIRO, Département d'Informatique et de Recherche Opérationnelle, local 2377
Http : //www.iro.umontreal.ca/∼mignotte/ift1214/
E-mail : mignotte@iro.umontreal.ca

Date : *26/04/2005*

I	Assembleur/LMC/Registres (55 pts).
II	Programmation Shell Script (30 pts).
III	Séquenceur Microprogrammé (10 pts).
IV	Misc. (20 pts).

Directives

- TOUTE DOCUMENTATION ET CALCULATRICE PERSONNELLE EST PERMISE

- Les réponses devront être clairement présentées et justifiées (elles peuvent être concises mais devront néanmoins contenir les résultats intermédiaires nécessaires permettant de montrer sans ambiguïté que vous êtes arrivés au résultat demandé).

- Si vous ne comprenez pas une question, faites en une interprétation, et proposez une réponse.

I. Assembleur/LMC/Registres (55 pts)

1. **Programmation de l'approximation de** $\exp 1$
   Une des façons utilisée par les microprocesseurs modernes pour approximer la valeur de fonctions particulières (tel que exp, sin, cos, etc) en certain points est d'utiliser le développement en série de Taylor et MacLaurin. La valeur de ces fonctions en un point particulier est ainsi approximer par une série polynomiale avec un nombre fini de termes (n'utilisant que les opérations élémentaires du CPU, i.e., addition, soustraction, multiplication et division). Ainsi, $\exp 1$ (ou $e \approx 2.7182818$) peut être approximé, par exemple, par la somme de ces $n = 5$ termes suivants :

   $$\exp 1 \quad \approx \quad 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} \approx 1 + \frac{1}{1} + \frac{1}{2} + \frac{1}{6} + \frac{1}{24}.$$

   Implémenter en code d'instructions mnémoniques, avec le jeux d'instruction donné plus bas correspondant au modèle LMC vu en cours, un programme permettant de calculer une valeur approximée de $\exp 1$ avec un nombre fini de $n \quad (> 1)$ termes. On supposera que la valeur de n est entrée au début du programme par l'instruction IN et que le résultat est communiqué en sortie par l'instruction OUT[1]. **<15 pts>**

   On utilisera le jeux d'instruction présenté dans le chapitre Little Man Computer (cf. Fig.) et auquel on ajoutera les deux instructions suivantes :

   (a) **MUL** xx Multiplie le contenu de la case mémoire xx au nombre stocké dans l'accumulateur (registre du calculateur). Le résultat de la multiplication est stocké dans l'accumulateur.

   (b) **DIV** xx Divise le contenu de la case mémoire xx au nombre stocké dans l'accumulateur (registre du calculateur). Le résultat de la division est stocké dans l'accumulateur.

LDA	5xx	Load
STO	3xx	Store
ADD	1xx	Add
SUB	2xx	Subtract
IN	901	Input
OUT	902	Output
COB or HLT	000	Coffee break (or Halt)
BRZ	7xx	Branch if zero
BRP	8xx	Branch if positive or zero
BR	6xx	Branch unconditional
DAT		Data storage location

2. **Série de nombres triangulaires**
   La suite $1, 3, 6, 10, 15, 21, \ldots$ est la suite des nombres dit triangulaires. Une façon de trouver ces nombres est de procéder comme suit :

$$\begin{aligned} 1 &= 1 \\ 3 &= 1+2 \\ 6 &= 1+2+3 \\ 10 &= 1+2+3+4 \\ 15 &= 1+2+3+4+5 \\ &\textit{etc.} \end{aligned}$$

1. On peut le réaliser rapidement en utilisant le pseudo code suivant,

$y \leftarrow 1;\quad inc \leftarrow 1;$
pour $k = 1$ à $(n-1)$
$\{\ inc \leftarrow inc \times \frac{1}{k};$
$y \leftarrow y + inc;\ \}$

Implémenter en code d'instructions mnémoniques (sans l'instruction MULT et DIV), avec le jeux d'instruction de base correspondant au modèle LMC vu en cours, un programme prenant une valeur d'entrée $n \geq 0$ et indiquant si ce nombre est triangulaire (dans ce cas la sortie doit indiquer sa position dans la série) ou si celui-ci ne l'est pas (dans ce cas la sortie doit indiquer 0). Exemple : si $n = 15$, la sortie devra être 5 (15 est le 5ième nombre triangulaire). Si $n = 7$, la sortie devra être 0 (7 n'est pas un nombre triangulaire). **<15 pts>**

3. **Langage LMC et registres**
Voici un programme en code d'instructions mnémoniques,

```
        IN
        STO   SMALL
        IN
        STO   BIG
        LDA   BIG
        SUB   SMALL
        BRP   OK
        LDA   SMALL
        STO   TEMP
        LDA   BIG
        STO   SMALL
        LDA   TEMP
        STO   BIG
OK      LDA   SMALL
        ADD   ONE
        STO   SMALL
LOOP    LDA   SMALL
        SUB   BIG
        BRP   END
        LDA   SMALL
        OUT
        ADD   ONE
        STO   SMALL
        BR    LOOP
END     HLT
ONE     DAT   1
SMALL   DAT   0
BIG     DAT   0
TEMP    DAT   0
```

(a) Indiquer en une ligne ce que ce programme va faire. Indiquer aussi ce qui se passe si on entre les séries de nombres suivant comme input à ce programme : **<5 pts>**

i. $(10, 20)$ (i.e., 10 pour le premier IN et 20 pour le second)

ii. $(20, 10)$

(b) Convertir les dix premières instructions de ce programme en Code machine (codes d'opérations) en logeant ce programme à partir de l'adresse mémoire 00. **<5 pts>**

(c) Donner la valeur des différents registres ACCU, IR, PC, MAR, MDR à la fin des instructions **STO BIG** (4 ième instruction) et **BRP OK** (7 ième instruction) pour les deux inputs suivant $(10, 20)$. **<5 pts>**

(d) On pourrait sauver un peu de temps machine en déplaçant une des deux LABELS de boucle (i.e., LABEL **OK** ou **LOOP**) dans ce programme. Dire laquelle et où devrait on déplacer cette LABEL pour que l'exécution du programme soit la même et qu'on puisse gagner un peu de temps d'exécution. **<5 pts>**
En supposant que chaque instruction s'effectue en un cycle d'horloge, indiquer en pourcentage le gain en calcul que cela permettrait d'obtenir pour une exécution du programme avec le couple d'inputs suivant $(10, 20)$. **<5 pts>**

Réponse

$\boxed{\textbf{1.}}$

Un exemple possible, valable $\forall\, n > 1$,

```
            IN
            SUB   UN
            STO   NBTERMS
LOOP        LDA   UN
            DIV   CPT
            MUL   INC
            STO   INC
            ADD   RES
            STO   RES
            LDA   CPT
            ADD   UN
            STO   CPT
            SUB   NBTERMS
            BRZ   END
            BR    LOOP
END         LDA   RES
            OUT
            HLT
UN          DAT   1
NBTERMS     DAT   0
CPT         DAT   1
RES         DAT   1
INC         DAT   1
```

<15 pts>

2.

Un exemple possible, valable $\forall\, n \geq 0$,

```
            IN
            STO   VALUE
LOOP        LDA   TRINUM
            SUB   VALUE
            BRP   ENDLOOP
            LDA   N
            ADD   ONE
            STO   N
            ADD   TRINUM
            STO   TRINUM
            BR    LOOP
ENDLOOP     LDA   VALUE
            SUB   TRINUM
            BRZ   EQUAL
            LDA   ZERO
            OUT
            BR    DONE
EQUAL       LDA   N
            OUT
DONE        HLT
VALUE       DAT   0
TRINUM      DAT   0
N           DAT   0
ZERO        DAT   0
ONE         DAT   1
```

<15 pts>

3a.

Ce programme va demander l'entrée de deux nombres et affichera à l'écran tous les entiers compris entre ces deux nombres, les bornes étant exclues.

Pour $(10, 20)$, le programme affichera donc $11, 12, 13, 14, 15, 16, 17, 18, 19$.

Pour $(20, 10)$, le programme affichera aussi $11, 12, 13, 14, 15, 16, 17, 18, 19$.

<5 pts>

3b.

Les dix premières conversions en code machine est immédiate,

```
00   901
01   326
02   901
03   327
04   527
05   226
06   813
07   526
08   328
09   527
...  ...
```

<5 pts>

2c.

	ACCU	IR	PC	MAR	MDR
Fin de **STO BIG**	20	327	04	27	20
Fin de **BRP OK**	10	813	13	06	813
	<1 pt>	**<1 pt>**	**<1 pt>**	**<1 pt>**	**<1 pt>**

2d.

On pourrait mettre le LABEL **LOOP** à l'instruction suivante, i.e. **LOOP SUB BIG**.

<5 pts>

Avec le couple d'inputs suivant $(10, 20)$

• et sans modification, le programme nécessiterait l'exécution de $10 + 8 \times 9 + 1 = 82$ instructions.

• avec modification, le programme nécessiterait l'exécution de $11 + 7 \times 9 + 1 = 75$ instructions.

En supposant que chaque instruction s'effectue en un cycle d'horloge, le programme se déroulera donc avec un gain en vitesse de 8.5%.

<5 pts>

II. Programmation Shell Script (30 pts)

1. **Manipulation/Modification de Fichiers**

   Voici un fichier de notes (appelé LISTE avec des noms d'étudiants fictifs) dans le format utilisé par l'administration pour la gestion des notes des étudiants.

```
:NOM Prenom:CODE               :.Tp1 :.Intra:.Tp2 :.Tp3 :.Final :.TP4 :.Moy:.Rang

actarusx:Actarus :ACTA28658105 :  45.5:  75.00:100.0:  95.0:  48.50:  90.0:  66.687 :
alcorxxx:Alcor   :ALCO18083408 :  83.0:  64.75:  95.0:  87.5:  74.00:  97.0:  76.207 :
venusiax:Venusia :VENU13097708 :  83.0:  55.00:  95.0:  87.5:  41.50:  97.0:  60.287 :
phenicia:Phenicia:PHEN14100790 :  94.0:  50.50:  95.0:  95.0:  59.50:  99.0:  67.675 :
mizarxxx:Mizar   :MIZA22128106 :  98.0:  95.50:  95.0:100.0:  80.50:  90.0:  89.575 :
procyonx:Procyon :PROC01033701 :  99.0:  99.50:100.0:  95.0:  98.00:100.0:  98.600 :
flamxxxx:Flam    :FLAM26077904 :  89.0:  37.25:  97.5:  87.5:  63.50:  97.0:  64.394 :
johannxx:johann  :JOHA23058000 :  94.0:  99.25:100.0:  95.0:  36.00:100.0:  73.345 :
wrightxx:Wright  :WRIG21058103 :  99.0:  100.0:100.0:100.0:  100.0:  97.0:  99.700 :
cragxxxx:Grag    :CRAG05018002 :  99.0:  90.00:  90.0:100.0:  97.00:  99.0:  94.900 :
malaxxxx:Mala    :MALA09557914 :  96.0:  69.50:100.0:100.0:  34.00:  47.0:  60.175 :
```

   (a) On vous demande de créer un script LISTORD.SH permettant d'obtenir la liste par ordre décroissant des moyennes des étudiants. On vous demande aussi de sauver cette liste de notes dans le fichier file.tmp (une note par ligne). Ce script devra passer en paramètre le fichier LISTE (pour une exécution du style ListOrd.sh LISTE et, dans le cas de ce fichier LISTE, donner un affichage dans le fichier file.tmp du style ;

99.700
98.600
94.900
...
60.175

et devra bien sur fonctionner pour tout autre fichier utilisant le même format[3].   <**12 pts**>

(b) On vous demande de créer un script Rang.sh permettant de donner le rang (i.e., le classement) de chaque étudiant. Ce script devra passer en paramètre le fichier LISTE (pour une exécution du style Rang.sh LISTE et créer le fichier LISTE-Rang qui aura la forme suivante :

```
:NOM Prenom:CODE              :.Tp1 :.Intra:.Tp2 :.Tp3 :.Final :.TP4 :.Moy:.Rang
actarusx:Actarus :ACTA28658105 : 45.5: 75.00:100.0: 95.0: 48.50: 90.0: 66.687 : 8/11:
alcorxxx:Alcor :ALCO18083408 : 83.0: 64.75: 95.0: 87.5: 74.00: 97.0: 76.207 : 5/11:
venusiax:Venusia :VENU13097708 : 83.0: 55.00: 95.0: 87.5: 41.50: 97.0: 60.287 : 10/11:
phenicia:Phenicia:PHEN14100790 : 94.0: 50.50: 95.0: 95.0: 59.50: 99.0: 67.675 : 7/11:
mizarxxx:Mizar :MIZA22128106 : 98.0: 95.50: 95.0:100.0: 80.50: 90.0: 89.575 : 4/11:
procyonx:Procyon :PROC01033701 : 99.0: 99.50:100.0: 95.0: 98.00:100.0: 98.600 : 2/11:
flamxxxx:Flam :FLAM26077904 : 89.0: 37.25: 97.5: 87.5: 63.50: 97.0: 64.394 : 9/11:
johannxx:johann :JOHA23058000 : 94.0: 99.25:100.0: 95.0: 36.00:100.0: 73.345 : 6/11:
wrightxx:Wright :WRIG21058103 : 99.0: 100.0:100.0:100.0: 100.0: 97.0: 99.700 : 1/11:
cragxxxx:Grag :CRAG05018002 : 99.0: 90.00: 90.0:100.0: 97.00: 99.0: 94.900 : 3/11:
malaxxxx:Mala :MALA09557914 : 96.0: 69.50:100.0:100.0: 34.00: 47.0: 60.175 : 11/11:
```

Ce script devra bien sur fonctionner pour tout autre fichier utilisant le même format mais aussi avec un nombre d'étudiants différents[4].   <**13 pts**>

2. **Recherche de Fichiers**
   On vous demande de créer un script spy.sh permettant de :
   Enregistrer dans un fichier (ListeMp3.dat) la liste des noms et chemin (en fait toute les informations que l'on pourrait obtenir comme avec la commande ls -al) de tous les fichier d'extension .mp3 existant dans son répertoire et sous répertoire et sous-sous répertoire etc. (où est executé le script). <**5 pts**>

Réponse

$\boxed{\textbf{1a.}}$

---

3. Un format avec lequel la moyenne des étudiants sera toujours en 10 ième champ. On supposera aussi qu'il y a toujours une note associée a chaque étudiant (pas d'ABSent) et qu'il y a le bon nombre d'arguments à ce script. Aide : On pourra stocker la liste des moyennes non classées dans un fichier temporaire que l'on pourra appeler tmp.dat et utiliser ce fichier pour créer le le fichier file.tmp final.

4. On supposera aussi qu'il y a toujours une note associée a chaque étudiant (pas d'ABSent) et qu'il y a le bon nombre d'arguments à ce script. On pourra efficacement se servir dans ce script de ce qui a été réalisé à la première question et appelé dans Rang.sh, le script ListOrd.sh. Aide : Vous pourriez utiliser la commande grep. Lorsqu'on sollicite la commande man grep, on a une option de cette commande qui peut s'avérer intéressante :
-n, --line-number
"Prefix each line of output with the line number within its input file."
Exemple : avec le fichier file.tmp de la question précédente et une commande du type :
grep -n "98.600" file.tmp
on obtient la réponse suivante :
> 2 : 98.600

```
#! /bin/sh
# ListOrd.sh script

# Recuperation Nom du fichier passe en parametre
InputFilename=$1

# Calcul du Nombre d'etudiants
NbEtudiants=`cat $InputFilename | wc -l`
let NbEtudiants=NbEtudiants-1

# Efface tmp.dat si existe
if [ -f tmp.dat ]
   then
   rm tmp.dat
fi

# Sauve les moyennes dans tmp.dat
cpt=1
while [ $cpt -le $NbEtudiants ]
do
let cpt=cpt+1
line=`head -n $cpt $InputFilename | tail -n 1`
Moyenne=`echo $line | cut -d: -f10`
echo $Moyenne >> tmp.dat
done

# Trie/affiche/sauvegarde liste des moyenne
cat tmp.dat | sort -r > file.tmp
cat file.tmp
```

<12 pts>

BARÈME DES 12 PTS :
Récup. de l'argument : 2 Pts
Récup. du nb. etudiants/Nb lignes : 2 Pts
Extraction de chaque Moy : 3 Pts
Trie/Affichage/Sauvegarde : 3 Pts
Reste : 2 Pts

Il y a aussi la version ultra-short pour gens pressés qui fonctionne tout aussi bien, i.e., la commande

```
cat LISTE | cut -d : -f 10 | tr ".Moy" " " | sort -r > file.tmp
```

1b.

```
#! /bin/sh
# Bang.sh script

# Input/Output
InputFilename=$1
OutputFile=${1}-BANG

# Calcul Nb d'etudiants
NbEtudiants=`cat $InputFilename | wc -l`
let NbEtudiants=NbEtudiants-1

# Ecriture 1ere ligne
head -n 1 $InputFilename > $OutputFile

# Appel ListOrd.sh
ListOrd.sh $1

# Afichage du Bang
cpt=1
while [ $cpt -le $NbEtudiants ]
do
let cpt=cpt+1
line=`head -n $cpt $InputFilename | tail -n 1`
Moyenne=`echo $line | cut -d: -f10`
Bang=`grep -n $Moyenne file.tmp | cut -d: -f1`
echo $line ${Bang}/${NbEtudiants}: >> $OutputFile
done
```

<13 pts>

Barème des 13 Pts :
Récup. de l'argument : 2 Pts
Récup. du nb. étudiants/Nb lignes : 2 Pts
Écriture 1ère ligne : 1 Pt
Liste Ord Moyenne : 2 Pts
Affichage Rang : 5 Pts
Reste : 1 Pt

2.

Deux versions courtes possibles :

```
find -name \*.mp3 -exec ls -al \{\} \; > ListeMp3.dat
```

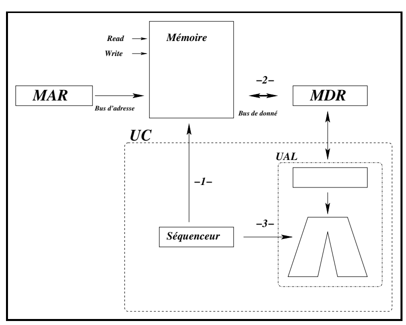
<5 pts>

```
find -name \*.mp3 | xargs ls -al > ListeMp3.dat
```

III. Séquenceur Microprogrammé (10 pts)

1. Quelles sont les avantages/inconvénients d'un séquenceur micro-programmés sur un séquenceur câblé. **<3 pts>**
2. Soit le modèle très simplifie de l'unité centrale d'un système informatique avec sa mémoire et les trois registres de base (MAR, MDR et ACC). Chacun de ses trois registres, y compris la mémoire, sont commandés par deux commandes Read et Write. Après avoir préciser quel serait le contenu de ces trois registres après la phase Fetch des commandes **LDA xx** et **STO xx**, Donner la succession des mots de 8 bits qui dans un séquenceur Microprogrammé permettrait de simuler la phase exécution de la commande **LDA xx** et **STO xx** avec le format de ces mots donné en Figure ci dessous. **<7 pts>**

Read	Write	Read	Write	Read	Write	Read	Write
Memoire		MDR		MAR		ACC	



Réponse

1.

L'avantage d'un séquenceur micro-programmé réside 1- dans sa souplesse et sa simplicité (on peut rajouter des instructions au jeux d'instructions très facilement ou émuler un jeux d'instruction d'une autre machine très facilement). 2- Cela simplifie considérablement l'architecture du CPU (le CPU est allégé du câblage logique des instructions). L'inconvénient est 1- qu'il est plus lent qu'un séquenceur câblé. <**3 pts**>

2.

LDA xx : Après la phase Fetch, le registre d'adresse MAR contient xx, les deux autres registres ont des valeurs indéterminés. <**1 pt**>

Pour la phase exécution de cette commande, le contenu de la case mémoire d'adresse xx doit aller dans le registre de l'accumulateur. A cette fin, et selon ce modèle (simplifié), cela peut être fait après la succession des mots suivants :

10 01 00 00 (xx) $\rightarrow$ MDR
00 10 00 01 MDR $\rightarrow$ ACC <**2.5 pts**>

Nota : On pourrait préceder ces deux mots par le mot 00001000 pour le transfert du CP au MAR.

STO xx : Après la phase Fetch, le registre d'adresse MAR contient xx, les deux autres registres ont des valeurs indeterminés. <**1 pt**>

Pour la phase exécution de cette commande, le contenu du registre ACC doit aller dans la case mémoire d'adresse xx. A cette fin, et selon ce modèle (simplifié), cela peut être fait après la succession des mots suivants :

00 01 00 10 ACC $\rightarrow$ MDR
01 10 00 00 MDR $\rightarrow$ (xx) <**2.5 pts**>

Nota : On pourrait préceder ces deux mots par le mot 00001000 pour le transfert du CP au MAR.

IV. Misc (20 pts)

1. **Paging**

   (a) Soit un système informatique pour lequel une instruction prend 1 nanoseconde (ns) et un *défaut de page*, prend un temps additionnel (à ces 1 ns) de n ns. Quel est le temps d'exécution effective d'une instruction si un *d*éfaut de page arrive tous les k instructions ? <**3 pts**>

   (b) Supposons qu'un petit système informatique ait une capacité de 4 pages mémoire. Un programme, qui s'exécute dans ce programme a besoin de 8 pages mémoire. Supposons que lors de son déroulement, ce programme ait besoin respectivement de la page numéro 0-1-7-2-3-2-7-1-0-3 (fin du programme). Calculer le nombre de défaut de page que va faire ce programme si le système de remplacement de page suit le principe d'une pile FIFO ou d'une pile LRU (Least Recently Used : i.e., dans cette stratégie, la page la moins récemment utilisée est remplacée). Justifiez votre réponse <**6 pts**>

2. **Cache**

   (a) Soit le système informatique suivant :
   — Cache C1 : Tps d'accès=7ns : Hit Ratio=0.80
   — Cache C2 : Tps d'accès=15ns : Hit Ratio=0.92
   — Mémoire : Tps d'accès=90ns : Hit Ratio=1.0

      i. Calculer le temps d'accès moyen pour une hiérarchie mémoire avec un seul niveau de cache (mémoire principale et cache C1). <**2 pts**>

      ii. Calculer le temps d'accès moyen pour une hiérarchie mémoire avec un seul niveau de cache (mémoire principale et cache C2). <**2 pts**>

iii. Calculer le temps d'accès moyen pour une hiérarchie mémoire avec deux niveaux de cache [mémoire principale + cache C1 (la plus près du processeur) + cache C2]. <**2 pts**>

3. **TéléInformatique**
   (a) Dire en une ou deux phrases quel est la différence entre le Web et Internet ? <**2.5 pts**>
   (b) Dire rapidement à quoi sert la modulation dans la transmission d'un signal ? <**2.5 pts**>

Réponse

1a.

Soit donc le système pour lequel on a ; $k-1$ instructions prenant chacune d'entre elles 1 ns et la k ième instruction prenant $(1+n)$ ns, cela nous donne une moyenne ou un temps d'exécution effective pour une instruction de $[(k-1)*1+(1+n)*1]/k$, i.e.,

$$(k+n)/k \qquad < \mathbf{3\ pts} >$$

1b.

(1) FIFO

0	1	7	2	3	2	7	1	0	3
0	1	7	2	3	3	3	3	0	0
	0	1	7	2	2	2	2	3	3
		0	1	7	7	7	7	2	2
			0	1	1	1	1	7	7
P	P	P	P	P				P	

<**3 pts**>

(2) LRU

0	1	7	2	3	2	7	1	0	3
0	1	7	2	3	2	7	1	0	3
	0	1	7	2	3	2	7	1	0
		0	1	7	7	3	2	7	1
			0	1	1	1	3	2	7
P	P	P	P	P				P	P

<**3 pts**>

Donc respectivement 6 et 7 défaut de page.

2a.

- $(7*0.8)+(0.2*90) = 23.6$ ns <**2 pts**>
- $(15*0.92)+(0.08*90) = 21$ ns <**2 pts**>
- $(7*0.8)+((0.2*0.92)*15)+(0.016*90) = 9.8$ ns <**2 pts**>

J'ai compté juste aussi pour le raisonement qui conduirait aux calculs suivants :
- $(7*0.8)+(0.2*(7+90)) = 25$ ns
- $(15*0.92)+(0.08*(15+90)) = 22.2$ ns
- $(7*0.8)+((0.2*0.92)*(7+15))+(0.016*(7+15+90)) = 11.44$ ns

3a.

Internet représente physiquement le réseau des réseaux (i.e., l'ensemble des connections). Le Web (ou World Wide Web) est la couche logicielle, application d'internet ou si vous preferrez l'ensemble des pièces logicielles (documents d'hypertextes) distribué par le réseau mondial internet et qui facilitent grandement son utilisation. <**2.5 pts**>

3b.

La modulation est le codage de l'information nécessaire à l'adaptation du message au canal de transmission. <**2.5 pts**>