VARI-Exercice Dirigé

Système d'exploitation

Corrigé indicatif

Exercice 1 - Chaîne de production d'un programme

Question 1

Rappeler les différentes étapes de la chaîne de production d'un programme.

Expliquez le rôle de chaque étape et le résultat obtenu.

La chaîne de production de programmes est un ensemble d'outils qui, à partir du texte d'un programme en langage d'assemblage ou en langage évolué fournit un *programme exécutable* pour une machine donnée.
Un programme exécutable est écrit en binaire, ses instructions appartiennent au jeu d'instructions de la machine, son espace d'adressage est inclus dans l'espace d'adressage mémoire de la machine.

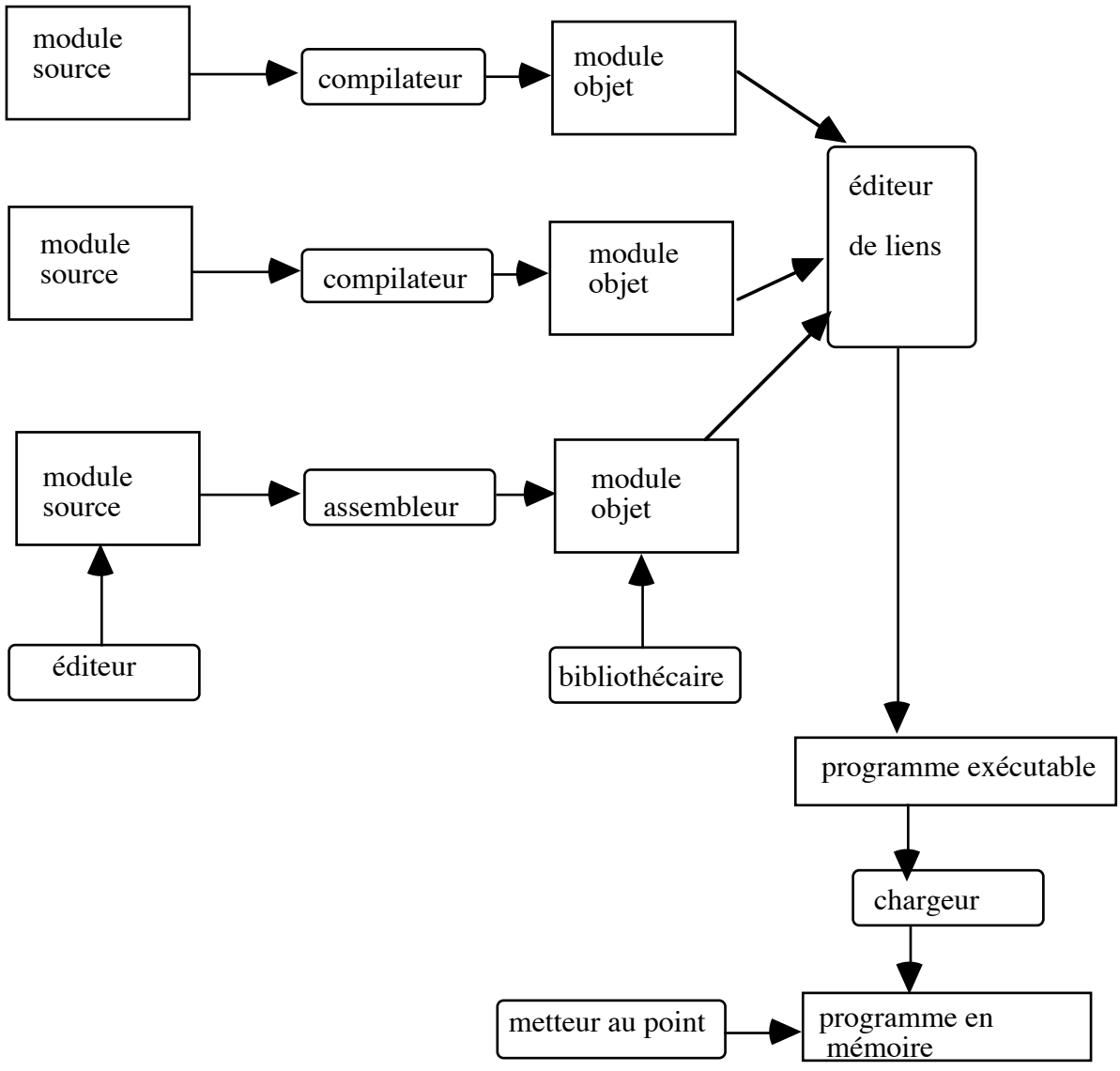
1ère étape L'édition de texte
L'éditeur de texte est un logiciel interactif permettant la création, la modification du texte des programmes. Le résultat fourni est un fichier texte, en général conservé sur disque, on parle de *module source*.
Afin de faciliter la tâche du programmeur, les éditeurs actuels permettent de tenir compte du langage utilisé et notamment des structures spécifiques. On les appelle des éditeurs syntaxiques.

2ème étape La traduction en langage machine
Les compilateurs (langage évolué) ou *les assembleurs* (langage d'assemblage) traduisent les symboles des instructions en codes binaires et les symboles des adresses en adresses de la machine.
Au préalable, le traducteur aura vérifié la correction des mots du texte (*analyse lexicale*), celle des phrases (*analyse syntaxique*) et certaines propriétés des objets du programme (*analyse sémantique*).
Le résultat est un programme binaire appelé *module objet*.
Associée au module objet, *la table des symboles* contient la liste des identificateurs (symbole d'adresse) et leur adresse correspondante dans le module.
Il existe également une autre catégorie de traducteurs, *l'interpréteur* qui analyse, traduit et exécute immédiatement chaque instruction.

3ème étape L'édition de liens
Quand une application devient importante, il est nécessaire de la découper en modules, chacun des modules étant compilé séparément. D'autre part, le programmeur peut être amené à utiliser des modules réalisant certaines fonctions, qui sont conservés dans des bibliothèques.
Le programme fait donc des références à des adresses symboliques qui ne sont pas dans son espace d'adressage. Le rôle de *l'éditeur de liens* est d'établir les adresses de ces références.

Auparavant, l'éditeur de liens aura constitué un espace d'adressage unique pour l'ensemble des modules.

4ème étape Le chargement en mémoire
Le chargeur place à partir d'une adresse mémoire, le module unique résultant de l'édition de liens, en mémoire centrale. Le programme peut alors être exécuté.



Etude de l'assembleur

Question 2

Complétez le tableau ci-dessous représentant les enregistrements du module objet.

Lors de l'assemblage (ou la compilation) d'un programme on ne peut générer des adresses mémoire définitives. Toutes les adresses du programmes seront exprimées par rapport au premier emplacement programme. Ce sont des adresses relatives.

type	nom ou adresse relative	taille ou code ou valeur	Texte
entête	ESSAI	6	NOM ESSAI
code absolu	0	25	A : MOT =25
code relatif	2	10000	LOAD A
code absolu	3	30010	MUL 10
code absolu	4	20001	ADD 1
code relatif	5	40001	STORE B
	fin_module		FIN

Remarques

1 le terme code relatif peut être remplacé par code relogeable, code à translater

2 B est une variable (donnée) non initialisée, on réserve seulement un emplacement à l'adresse 1, il n'y a donc pas d'enregistrement pour B.

Etude de l'éditeur de liens

L'éditeur de liens constitue un module unique en plaçant les différents modules les uns à la suite des autres. Il faudra pour toutes les adresses d'un module (code relatif), ajouter à ces adresses l'adresse du début du module, dans le module unique.
La deuxième étape consiste à résoudre les liens, c'est-à-dire affecter une adresse pour toute référence du programme à un module extérieur.

Question 3

Modifier le module objet issu de l'assemblage du programme ci-dessus, en complétant le tableau :

type	nom ou adresse relative	taille ou code ou valeur	Texte
entête	ESSAI	7	NOM ESSAI
lien utilisable	INCR	2	PUBLIC
lien à satisfaire	SUITE		EXTERN
code absolu	0	25	A : MOT =25
code relatif	2	10000	LOAD A
code absolu	3	30010	MUL 10
code absolu	4	20001	ADD 1
code relatif	5	40001	STORE B
code avec lien à satisfaire	6	* 50000	JMP SUITE
	fin module		FIN

Remarque * 50000 on note que l'adresse est inconnue, lien à satisfaire par l'éditeur de liens, on ne traduit que le code opération

Question 4

Donnez les enregistrements fournis par l'éditeur de liens pour le module ESSAI en complétant le tableau ci dessous.

type	adresse	code ou valeur	Texte
code absolu	123	25	A : MOT =25
code relatif	125	10123	LOAD A
code absolu	126	30010	MUL 10
code absolu	127	20001	ADD 1
code relatif	128	40124	STORE B
code relatif	129	50138	JMP SUITE

Etude du chargeur

Question 5

Donnez le contenu des emplacements mémoire contenant le module ESSAI.

1165	25	A=25
1166	?	B
1167	11165	LOAD A
1168	30010	MUL 10
1169	20001	ADD 1
1170	41166	STORE B
1171	51180	JMP SUITE

Exercice 2 : Exemple d'éditions de liens

L'adresse d'implantation d'un module est 0 pour le premier, et pour les autres, le premier emplacement laissé libre par le module qui le précède, c'est-à-dire, la somme entre son adresse d'implantation et sa taille:

```
   PROGRAMME         0
   ETIQUETTE       332
   LECTURE         460
   IMPRESSION     1300
   EDITION        1512
```

La taille totale du programme est la somme des tailles de tous les modules, ou encore l'adresse du premier emplacement laissé libre par le dernier module, c'est-à-dire, la somme entre son adresse d'implantation et sa taille, soit 2154.

La table des liens contient tous les identificateurs présents dans les modules, en tant que lien utilisable ou lien à satisfaire, ainsi que éventuellement les noms des modules. De plus, la table associe à tout identificateur rencontré comme lien utilisable dans un module, leur adresse dans le programme, obtenue en ajoutant à leur adresse dans ce module l'adresse d'implantation du module. Enfin aux noms de modules, la table associe l'adresse d'implantation du module.

```
*  PROGRAMME         0
   OUVRIR          460
   LIRE            800
   FERMER         1192
   EDITER         1512
*  ETIQUETTE       332
   NOM             332
   SOCIETE         364
   ADRESSE         396
   CODEPOST        428
   VILLE           433
*  LECTURE         460
*  IMPRESSION     1300
   IMPRIMER       1300
*  EDITION        1512
```

Les lignes marquées par un "*" ne sont pas toujours dans la table, suivant que l'on utilise ou non la table pour mémoriser les adresses d'implantations des modules.

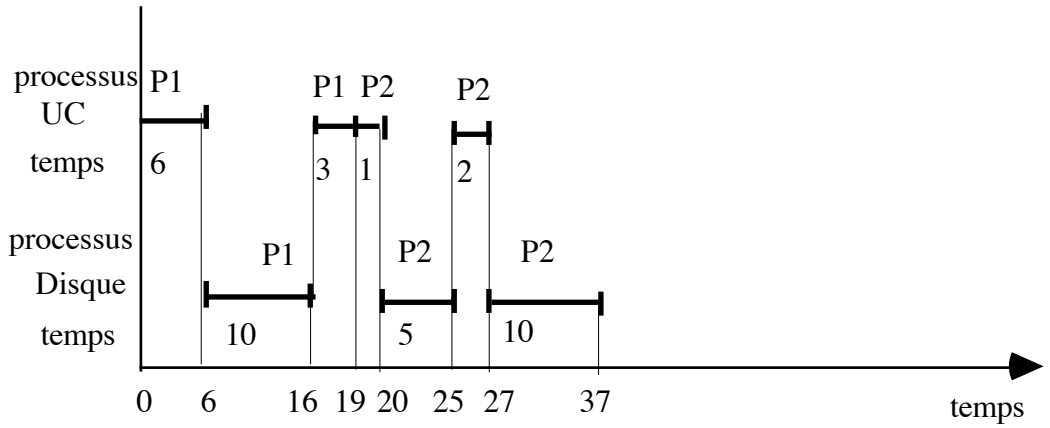
L'adresse de lancement est la valeur trouvée dans l'un des modules augmentée de l'adresse d'implantation de ce module: 133 + 0 = 133**.**

Exercice 3 : Processus en multiprogrammation.

Question 1

On lance et exécute P1, on lance ensuite immédiatement P2 et on l'exécute .

Quels sont les temps de réponse pour P1, pour P2 ? Quel est le temps global de réponse ?



t=0 lancement de P1, t=6 demande d'E/S de P1, t=16 IT fin d'E/S pour P1,

t=19 fin de P1, lancement de P2, t=20 demande d'E/S de P2, t=25 IT fin d'E/S pour P2, t=27 demande d'E/S de P2, t=37 IT fin d'E/S pour P2, fin de P2.

temps de réponse pour P1 : 19 ms, temps de réponse pour P2 : 18 ms,

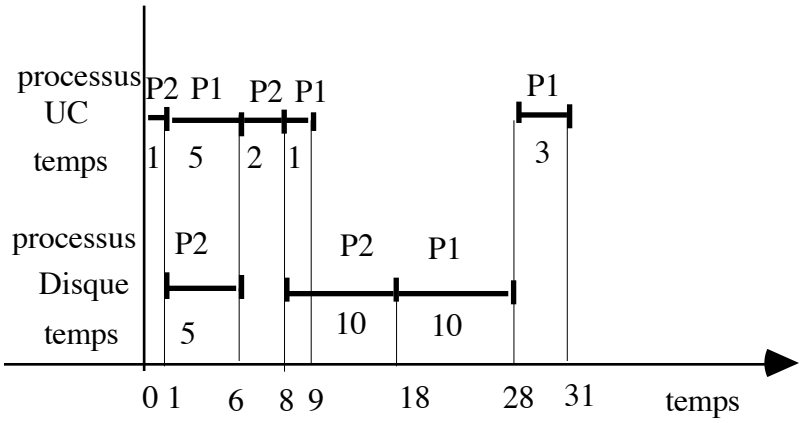
temps global de réponse : 37 ms.

NB On a négligé les temps de commutation de contexte et de traitement des interruptions

Question 2

Quels sont les temps de réponse pour P1, pour P2 ? Quel est le temps global de réponse ?

Comparez aux résultats obtenus dans la question 4 et commentez.



t=0 lancement de P1 et P2, priorité à P2

t=1 demande d'E/S de P2, exécution de P1

t=6 IT fin E/S pour P2 : P2 a priorité et s'exécute, attente de P1

t=8 demande d'E/S de P2, P1 s'exécute

t=9 demande d'E/S de P1, attente disque

t=18 IT fin E/S pour P2, fin P2

t=28 IT fin E/S pour P1

t=31 fin P1

temps de réponse pour P1 : 31ms, pour P2 : 18 ms, temps global : 31 ms

Par rapport à une exécution séquentielle, P1 a un temps de réponse augmenté, P2 le même temps de réponse (c'est dû à sa priorité), le temps global est amélioré.

Conclusion : la multiprogrammation améliore le débit global, mais le temps de réponse pour les usagers est augmenté.