esir
ECOLE SUPERIEURE D'INGENIEUR DE RENNES

UNIVERSITÉ DE RENNES I

ESIR 1 option informatique

SYSTÈMES

Règles de Programmation pour les TP système

Ce document présente un certain nombre de règles d'écriture des programmes que vous *devez* respecter dans la réalisation de vos TP de système.

Ces règles sont destinées à organiser la présentation et la documentation de vos programmes, pour en faciliter l'écriture et les tests, et pour rendre plus facile la réutilisation de certaines parties d'un TP à l'autre. Un outil est mis à votre disposition pour extraire automatiquement une documentation de votre texte source.

Le respect de ces règles est d'autant plus important que le langage utilisé est de bas niveau.

1 Organisation générale d'un module

Un **module** est un morceau de texte source soumis à l'assembleur. Un **programme** peut être constitué d'un ou de plusieurs modules.

Un module contient les parties suivantes (certaines parties peuvent être absentes) :

- partie **en-tête de module**, commentaires destinés à fournir des informations générales sur le module (obligatoire).
- partie **déclarations globales**, contenant les déclarations des objets utilisés dans l'ensemble du module (peut ne pas exister).
- partie **sous-programmes**, contenant la description des différents sous-programmes définis dans le module (peut ne pas exister). Pour chaque sous-programme on a :
  - un **en-tête de sous-programme**, commentaires destinés à fournir des informations sur le sous-programme (obligatoire).
  - des **déclarations locales**, contenant les déclarations des objets utilisés uniquement dans le sous-programme (peuvent ne pas exister).
  - les **instructions** du sous-programme (obligatoire).
- partie **programme principal** (peut ne pas exister).

2 Règles générales d'écriture

- Les directives et instructions seront présentées en alignant les colonnes correspondant aux différents champs : étiquette, code, arguments, commentaires.

  Le champ étiquette commence en colonne 1.

  Les étiquettes d'instructions sont isolées (seul un commentaire peut figurer en plus sur la ligne)

- Les identificateurs doivent être significatifs. On peut utiliser des majuscules et des minuscules pour les rendre plus lisibles, bien que l'assembleur ne fasse pas la différence.
- Dans un sous-programme tous les identificateurs (sauf le nom du sous-programme) sont préfixés par "XX_", où "XX" représente une ou plusieurs lettres spécifique du sous-programme, afin de faciliter la localisation de la déclaration et d'éviter les problèmes d'homonymies entre sous-programmes.

Exemple :

```
; SOUS-PROGRAMME LIRENT
...
            .DATA
LI_chu      db      ?           ; chiffre des unites
LI_chd      db      ?           ; chiffre des dizaines
...
lirent:
            push    ax          ; code du sous-programme LIRENT
...
            mov     LI_chd,ax
...
LI_fin:                         ; fin de LIRENT
...
```

3 En-tête de module

L'en-tête de module doit regrouper les informations permettant d'une part de déterminer, en phase de mise au point ou de modification, s'il y a besoin de rentrer dans une étude plus précise de ce module; et d'autre part de savoir ce que le module contient pour pouvoir le réutiliser.

Il doit respecter le modèle suivant, dans lequel :

- les caractères gras correspondent à des mots clés devant figurer tel quel dans l'en-tête.
- toutes les parties doivent figurer même si elles sont vides.

Modèle d'en-tête de module :

```
;*-------------------------------------------------------------
;*MODULE        nom du module
;*-------------------------------------------------------------
;* auteurs - date de dernière modification
;*-------------------------------------------------------------
;*OBJET
;*        description générale du rôle du module
;*POINTS d'ENTREES
;*        description des sous-programmes du module
;*        pouvant être appelés de l'extérieur
;*DONNEES EXPORTEES
;*        description des données définies dans le module et
;*        pouvant être utilisées dans un autre module
;*REFERENCES EXTERNES
;*        description des objets (sous-programmes ou données)
;*        utilisés dans le module et définies dans un autre
```

```
;*        module
;*-------------------------------------------------------------
;*SOUS-PROGRAMMES
;*        liste des sous-programmes définis dans le module
;*-------------------------------------------------------------
```

4 En-tête de sous-programme

L'en-tête de sous-programme doit contenir la spécification du sous-programme et la description précise de son mode d'appel. Il doit respecter le modèle suivant, avec les mêmes conventions que pour l'en-tête de module

Modèle d'en-tête de sous-programme:

```
;*-------------------------------------------------------------
;*SOUS-PROGRAMME              nom du sous-programme
;*-------------------------------------------------------------
;*SPECIFICATION
;*        description précise du rôle du sous-programme et
;*        de ses paramètres
;*PASSAGE DES PARAMETRES
;*        description précise du mode de passage des
;*        paramètres
;*-------------------------------------------------------------
;*MISE EN OEUVRE
;*        liste des sous-programmes et données globales
;*        utilisées dans le sous-programme
;*-------------------------------------------------------------
```

5 Présentation des déclarations

- Les déclarations doivent être structurées en regroupant et en isolant les déclarations de même "type", par exemple :
  - les constantes liées à l'application (taille des tables,...)
  - les constantes liées au système d'E/S (codes des fonctions,...)
  - les structures de données exportées
  - les structures de données locales au module
  - les messages pour affichage
  - etc.
- Chaque déclaration d'identificateur doit être commentée, sur la ligne de déclaration.

6 Documentation des instructions

- Les instructions doivent être commentées de manière synthétiques : sans paraphraser une instruction particulière on s'attachera à donner la signification globale d'un paquet d'instruction réalisant une action logique identifiable. Ces commentaires seront de préférence placés dans le champs commentaire des instructions.

Exemple :

```
mov   ah,lircar    ; lecture et decodage du
int   21h          ; chiffre des dizaines
and   al,Lmasq     ;
```

- Il n'est en général pas recommandé de reproduire un algorithme en commentaire, sauf dans ses grandes lignes.
- Pour une itération on doit facilement identifier :
  - le début et la fin des instructions répétées
  - les conditions de sortie de l'itération
- Dans le cas de traitement conditionnel on doit facilement identifier :
  - le début et la fin du traitement conditionnel
  - la condition pour qu'il soit exécuté

7 Outils d'extraction de la documentation

Le fichier **G:\esir1\ise\extract** contient un programme permettant d'extraire les en-têtes de module et de sous-programmes de fichiers contenant des textes source en assembleur.

Il demande à l'utilisateur les noms des fichiers à analyser et le nom du fichier résultat.

On peut demander 3 types de sortie pour chaque fichier analysé :

- l'en-tête de module (type de sortie *m* - "module")
- l'en-tête de module et les parties "spécification" et "mode d'appel" de chaque en-tête de sous-programme (type de sortie *r* - "réduit")
- l'en-tête de module et les en-têtes de sous-programme complets (type de sortie *c* - "complet")

Cette extraction ne se fait correctement que si les règles de présentation des en-têtes vues ci-dessus sont respectées. *En particulier il est nécessaire que les lignes des en-têtes commencent par la séquence ";*"*