Architecture des Ordinateurs L2 – Devoir Écrit - Corrigé

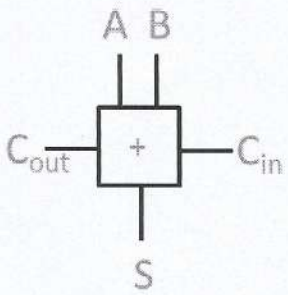
Durée : 2 heures. Aucun document ou appareil électronique n'est autorisé. Les exercices sont indépendants. Le barème est indicatif. Préciser clairement les numéros d'exercice et de question traités.

La correction apparait en gras, accompagnée d'explications. Celles-ci n'étaient pas demandées dans le devoir.

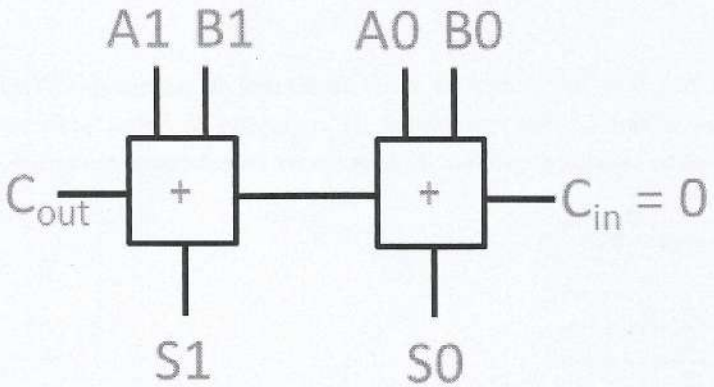
Exercice 1 – UAL 2 bits (5 points)

Cet exercice consiste à construire progressivement une UAL (Unité Arithmétique et Logique) simple, ayant deux entrées A et B de 2 bits chacune : A_1A_0 et B_1B_0.

1. On donne ci-dessous le schéma d'un additionneur 1 bit complet, avec retenue entrante C_{in} et sortante C_{out}. Indiquez comment construire à l'aide de ce type de circuit un additionneur complet pour deux opérandes 2 bits A_1A_0 et B_1B_0. Précisez la valeur de la retenue entrante de l'étage de poids faible.

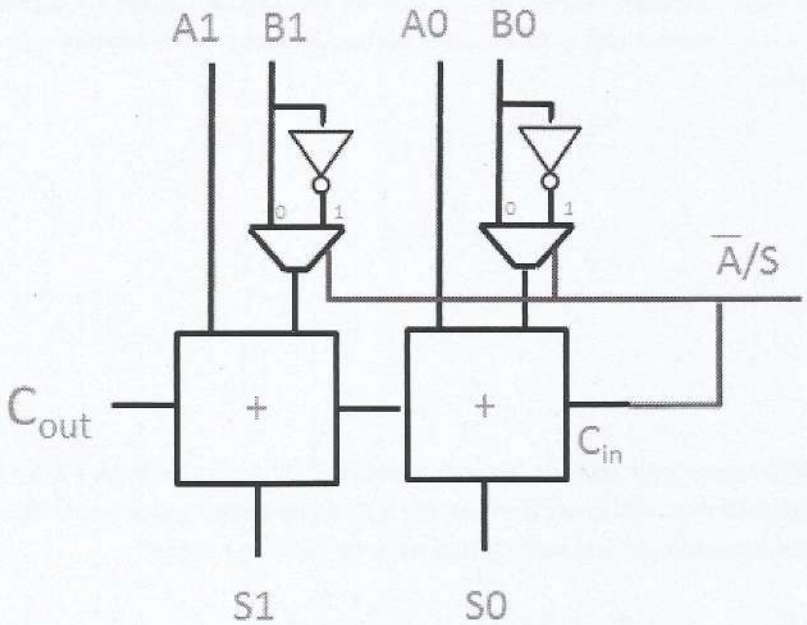


On met deux additionneurs 1 bit en série, chacun calculant 1 bit de la somme A + B. La retenue entrante de l'étage 0 (poids faible) est positionnée à 0, sa retenue sortante est connectée à l'étage 1. La retenue sortante globale de l'additionneur est celle de l'étage 1.



2. On souhaite maintenant compléter cet additionneur de façon à ce qu'il réalise également une soustraction en complément à 2, en fonction d'un signal d'entrée nommé $\overline{A}$ / S. Quand ce signal vaut 0, le circuit effectue l'addition A+B ; quand il vaut 1, il effectue la soustraction A-B. Indiquez comment construire ce circuit à partir du précédent, de multiplexeurs 2 vers 1 et de portes NOT.

Le circuit repose sur le fait que A – B = A + $\overline{B}$ + 1 en complément à 2. Pour chaque bit, on utilise un multiplexeur pour injecter en entrée de l'additionneur soit Bi, soit $\overline{Bi}$. La retenue entrante doit être 0 pour une addition et 1 pour une soustraction : elle est donc égale au signal $\overline{A}$ / S. Noter que la numérotation des entrées des multiplexeurs est importante pour la compréhension du schéma.

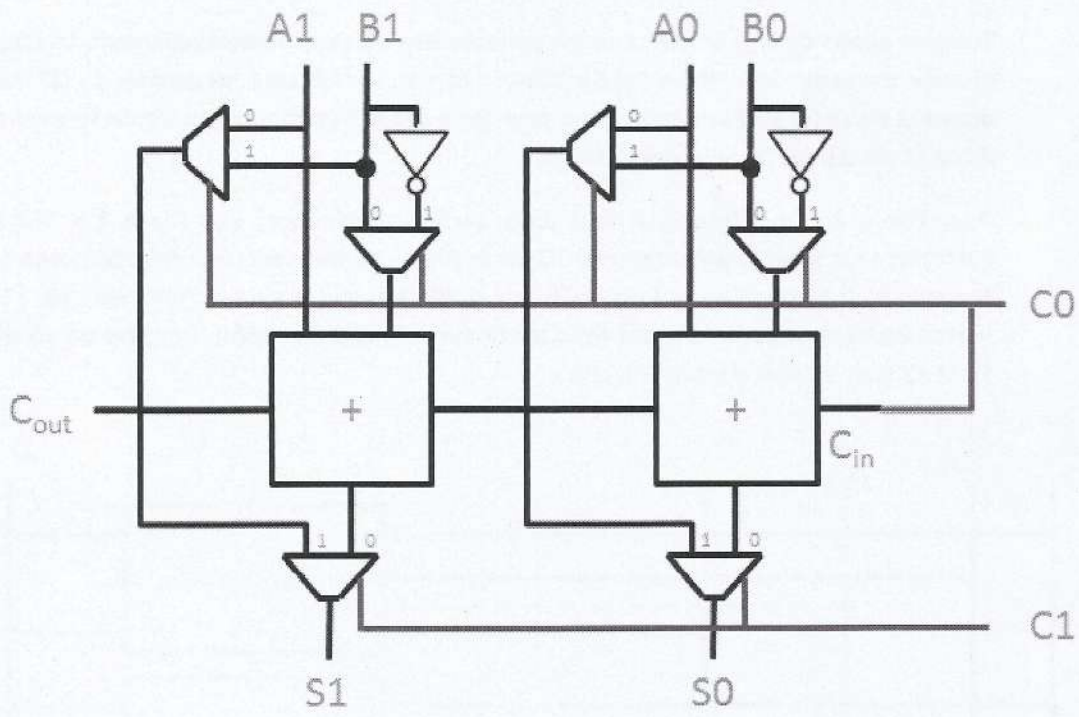


3. Une des fonctions d'une UAL est de retransmettre sur sa sortie une de ses entrées (A ou B) sans appliquer aucune transformation. Il s'agit maintenant de compléter le circuit précédent pour obtenir une UAL dotée de deux signaux d'entrée C_1C_0 et réalisant les fonctions suivantes :

C_1	C_0	Fonction UAL
0	0	S = A + B
0	1	S = A – B
1	0	S = A
1	1	S = B

Donnez le schéma de cette UAL. Vous pourrez par exemple utiliser deux multiplexeurs 2 vers 1 supplémentaires pour chaque bit : le premier sélectionnera une des deux entrées A ou B, le second sélectionnera une entrée ou le résultat de l'additionneur. Vous pourrez bien sûr utiliser toute porte logique supplémentaire dont vous auriez besoin.

Au vu de la table des fonctions, C_0 sert à sélectionner soit l'addition, soit la soustraction : il a un rôle similaire au signal $\overline{A}$ / S de l'exercice précédent. C_0 sert aussi à sélectionner soit A, soit B quand on ne fait pas d'opération : cette fonction sera réalisée via un deuxième multiplexeur. C_1 sert à sélectionner soit une opération (sortie de l'additionneur/soustracteur), soit un des opérandes (sortie du second multiplexeur) : cette sélection sera réalisée via un troisième multiplexeur.



Exercice 2 – Mémoire centrale (5 points)

On cherche dans cet exercice à implémenter une mémoire centrale à partir de barrettes mémoire standard du marché. On prend des barrettes de 128 Mo (= 2^{27} octets) constituées de mots de 8 bits. On considère sur chaque barrette les broches suivantes : broches adresses $A_{n-1}A_{n-2}...A_1A_0$, broches données $D_{m-1}D_{m-2}...D_1D_0$, et broche CS (Chip Select) indiquant si la barrette est concernée par un accès en lecture ou écriture. Dans le cadre de l'exercice, on ignore les autres broches telles que R / $\overline{W}$ et OE.

1. Indiquer le nombre de broches adresses (n) et le nombre de broches données (m) de chaque barrette.

   $n = \log_2 (128 \times 1024 \times 1024) = 27$ (numérotées 0 – 26)

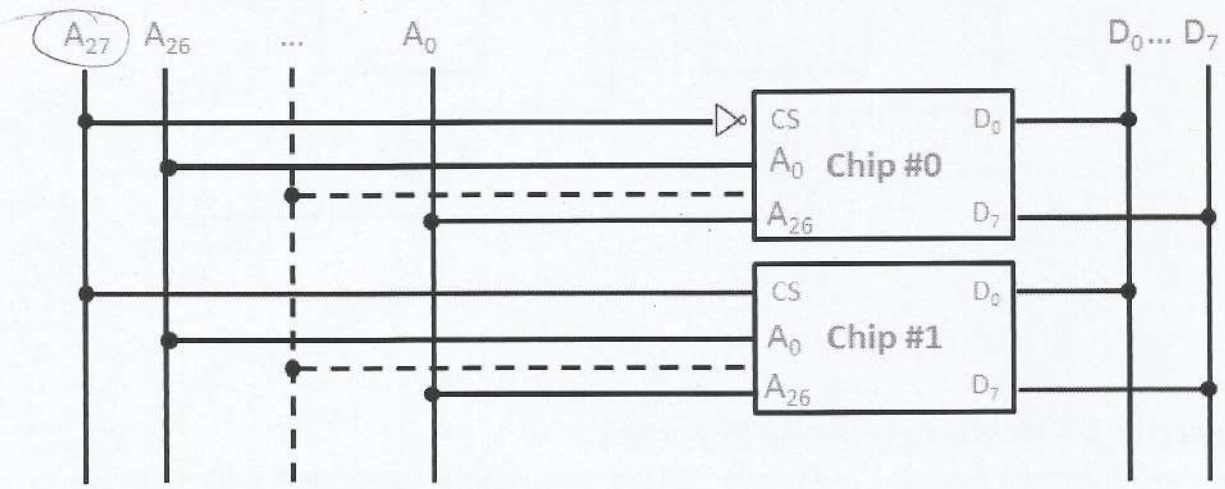
   $m = 8$ (numérotées 0 – 7)

A l'aide de ces barrettes, on veut dans un premier temps implémenter une mémoire centrale de 256 Mo ($=2^{28}$ octets) organisée en mots de 8 bits. On décide d'organiser les barrettes en banc.

2. Indiquez la largeur du bus adresses et du bus données de la mémoire centrale. Tracez son schéma d'implémentation à l'aide des barrettes standard et d'éventuelles portes logiques. Indiquez sur le schéma dans quelle(s) barrette(s) sont stockés l'octet d'adresse 10_{10} (valeur 0xAB) et celui d'adresse 11_{10} (valeur 0xCD).

   Tous les octets de la mémoire doivent pouvoir être adressés individuellement. La largeur bus adresse est donc : $\log_2 (256 \times 1024 \times 1024) = 28$ bits, ses fils sont numérotés 0 – 27. Le bus de données doit être suffisamment large pour faire circuler un mot mémoire. Sa largeur est donc 8 bits (1 octet), ses fils sont numérotés 0 – 7.

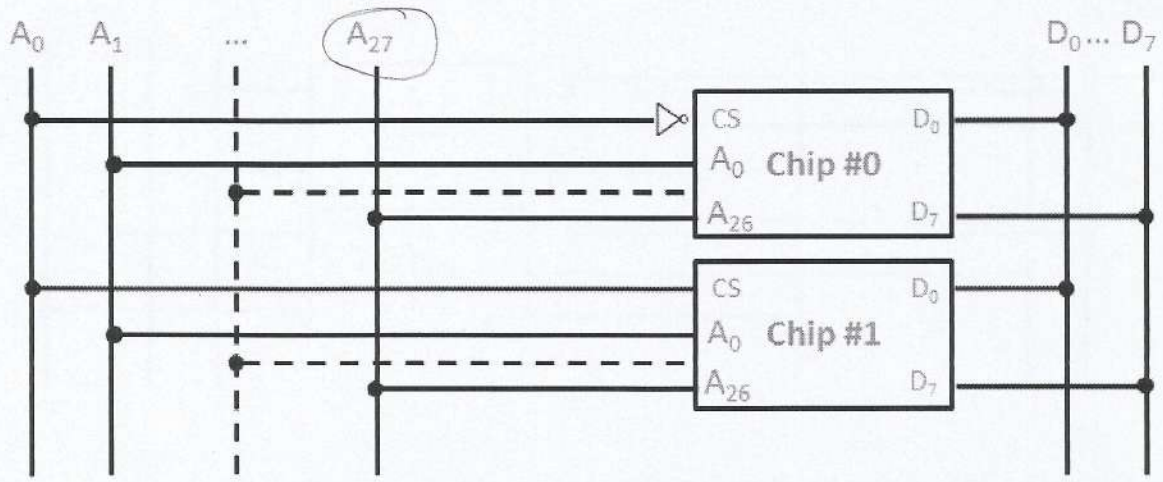
   Pour constituer la mémoire, il faut deux barrettes mémoire : 256 Mo = 2 x 128 Mo. Les barrettes sont sélectionnées (entrée CS) par le bit A_{27} du bus adresse (organisation en banc). La barrette n° 0 contient alors tous les octets dont l'adresse est dans l'intervalle [0, 128M[; la barrette n° 1 ceux dont l'adresse est dans l'intervalle [128M, 256M[. Ainsi les octets d'adresse 10 et 11 sont stockés dans la barrette n° 0.



On veut maintenant implémenter la même mémoire centrale (même taille totale, même taille de mot) mais en organisant cette fois les barrettes en mode entrelacé.

3. Reprenez la question 2 avec cette nouvelle hypothèse.

n, m et le nombre de barrettes restent les mêmes. Cette fois, les barrettes sont sélectionnées (entrée CS) par le bit A_0 du bus adresse (organisation entrelacée). La barrette n° 0 contient tous les octets dont l'adresse est paire ; la barrette n° 1 contient tous ceux dont l'adresse est impaire. Ainsi l'octet d'adresse 10 est stocké dans la barrette n° 0, et l'octet d'adresse 11 est stocké dans la barrette n° 1.



On veut enfin implémenter une mémoire centrale de 256 Mo, mais cette fois organisée en mot de de 2 octets. En plus des signaux d'adresse, le processeur émet un signal nommé B / $\overline{W}$ indiquant à la mémoire la taille de la donnée accédée. Lorsque ce signal vaut 0, le processeur accède au mot situé à l'adresse donnée ; quand il vaut 1, le processeur accède à l'octet situé à l'adresse donnée.

4. Indiquez la largeur du bus adresses et du bus données de la mémoire centrale. Tracez son schéma d'implémentation à l'aide des barrettes standard et d'éventuelles portes logiques, en prenant en compte le signal B / $\overline{W}$. Indiquez sur le schéma dans quelle(s) barrette(s) est stocké l'entier 0xABCD situé à l'adresse 10_{10} sachant que le processeur est de type *little endian*.

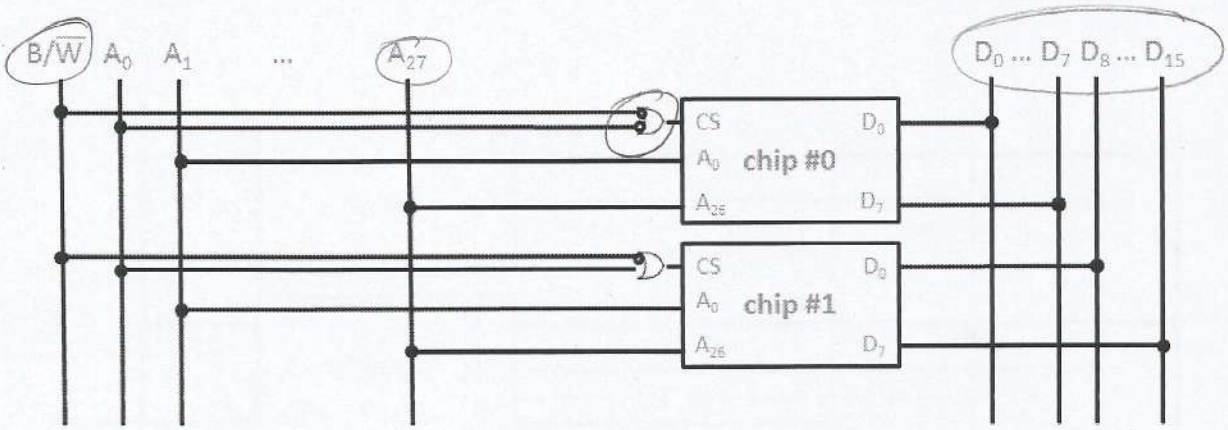
La taille en octet de la mémoire restant constante, la largeur du bus adresse reste la même. Celle du bus de données passe à 16 bits (mot de 2 octets), ses fils sont numérotés 0 - 15.

Il faut toujours deux barrettes mémoire : 256 Mo = 2 x 128 Mo. La barrette n° 0 contient l'octet d'adresse paire de chaque mot, la barrette n° 1 l'octet d'adresse impaire. Elles sont connectées respectivement à la première (D_{0-7}) et à la deuxième (D_{8-15}) moitié du bus de données.

Le processeur étant little endian, l'octet de poids faible d'un mot est stocké dans les adresses basses, l'octet de poids fort dans les adresses hautes. L'octet 0xAB (poids fort) de l'entier

0xABCD est donc stocké dans la barrette n° 1 et l'octet 0xCD (poids faible) est stocké dans la barrette n° 0.

La barrette n° 0 est sélectionnée (entrée CS) si on accède à un octet d'adresse paire (A_0 = 0) ou à un mot (B / $\overline{W}$ = 0). La barrette n° 1 est sélectionnée si on accède à un octet d'adresse impaire ou à un mot.



Exercice 3 – Mémoire cache (5 points)

On considère dans cet exercice un cache à correspondance directe. Il dessert une mémoire de 4 Go (=2^{32} octets), organisée en mots de 2 octets. Le cache est composé de 8 entrées ; chaque entrée contient une ligne de 4 mots.

1. Donnez la capacité du cache en octets, i.e. la taille des données utiles qu'il peut contenir, hors données de contrôle.

   Capacité cache = 8 entrées x 4 mots (/ entrée) x 2 octets (/ mot) = 64 octets

2. Indiquez comment le cache interprète une adresse mémoire émise par le processeur : positionnez et nommez ses différents champs, donnez leur largeurs et indiquez leur rôles.

   Les adresses émises par le processeur ont 32 bits. Quand il les reçoit, le cache les découpe et les interprète selon la structure suivante :

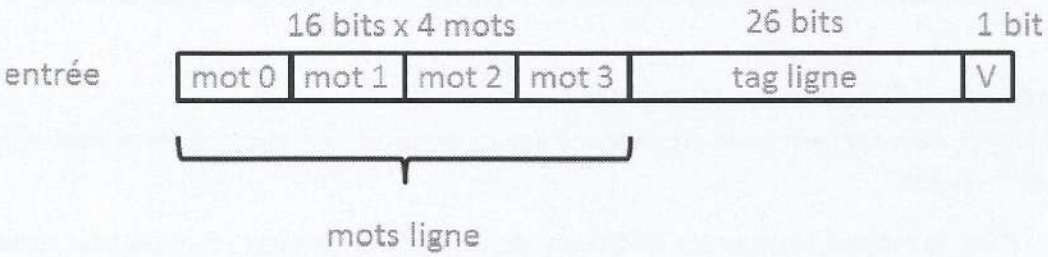
   L'adresse est composée de 4 champs. En partant des bits de poids faible, on a :

   - **byte : identifie un octet dans un mot ; largeur : $\log_2 2$ = 1 bit**

- **word : identifie un mot dans une ligne ; largeur : $\log_2 4 = 2$ bits**
- **tag et index : identifient une ligne en mémoire ; largeur : 32 – (1 + 2) = 29 bits**
  - **index : identifie l'entrée du cache dans laquelle sera stockée la ligne ; largeur : $\log_2 8$ (entrées) = 3 bits**
  - **tag : identifie une ligne d'index donné dans le cache (poids les plus forts de l'adresse) ; largeur : 29 – 3 = 26 bits**

3. Dessinez la structure d'une entrée du cache : nommez ses différents champs, donnez leur largeurs et indiquez leur rôles.

La structure d'une entrée du cache est la suivante :



Mots ligne : contient les différents mots de la ligne mémoire stockée dans cette entrée

Tag ligne : contient la partie « tag » de l'adresse de la ligne mémoire stockée dans cette entrée

V bit : indique si les informations des champs « mots ligne » et « tag ligne » sont valides

4. Comparez les caches associatifs et les caches à correspondance directe, en donnant leurs avantages et inconvénients.

Cache associatif :
- **Avantage : une ligne de mots peut être placée dans une entrée quelconque du cache. Il est donc possible d'implémenter une politique d'éviction, comme par exemple LRU. Ceci contribue à un taux de hit élevé et donc à une meilleure performance du cache.**
- **Inconvénient : pour être rapide, la recherche d'un mot dans le cache nécessite d'avoir un comparateur par entrée de cache. Le coût de ce type de cache est donc élevé.**

Cache à correspondance directe :
- **Avantage : une ligne de mots ne peut être placée que dans une seule entrée du cache : celle correspondant à son index. Il suffit donc d'un seul comparateur pour l'ensemble des entrées. Le coût matériel de ce type de cache est donc réduit.**
- **Inconvénient : puisqu'il n'est pas possible de choisir l'entrée dans laquelle on rapatrie une ligne de mots, aucune politique d'éviction ne peut être implémentée. Suite à un miss, une**

ligne va peut-être devoir être placée dans une entrée contenant une ligne très récente. Ceci conduit à un taux de hit plus faible et à des performances moindres.

5. Rappelez quelle propriété des programmes s'exécutant sur un ordinateur conduit à organiser un cache en *lignes* de mots. Donnez un exemple de cette propriété.

   C'est la propriété de localité spatiale : si un programme accède à une adresse mémoire, il est très probable qu'il accède à des adresses voisines très rapidement.

   Un exemple de cette propriété : lorsqu'un programme accède à une cellule d'un tableau, il est très probable qu'il accède très rapidement à des cellules voisines du tableau. C'est notamment le cas pour la plupart des algorithmes de tri, qui balayent un tableau du début à la fin.

Exercice 4 – Processeur (5 points)

On considère dans cet exercice le processeur à accumulateur vu en cours. Son jeu d'instructions est rappelé en annexe.

1. Dans un tableau, indiquez les différentes phases que le processeur enchaîne pour traiter une instruction, avec une ligne par phase. Dans deux colonnes, indiquez pour chaque phase (i) le nombre d'accès mémoire qu'elle induit en lecture (L) et/ou en écriture (E), (ii) les informations concernées par ces accès : instructions (I) et/ou données (D).

Phases	*Accès mémoire (L / E)*	*Informations (I / D)*
FETCH	**1 L**	**I**
DECODE	**0**	**-**
EXECUTE	**1 L ou 1 E**	**D**

2. Rappelez quelles instructions du processeur mettent à jour les bits du Condition Code Register (CCR), et quelles instructions les prennent en compte.

   Instructions mettant à jour les bits Z, N, C et V du CCR : instructions arithmétiques et logiques (ADD, SUB, MUL, DIV), instruction de comparaison (CMP), instruction de chargement (LDA).

   Instructions prenant en compte ces bits : instructions de branchement conditionnel (BRZ, BRN, BRC, BRV).

On considère le programme suivant, écrit en C. Il calcule la factorielle du nombre contenu dans la variable n, et range le résultat dans la variable f. Note : on considère que la variable n est initialisée avant le lancement du programme, mais qu'en revanche la variable f ne l'est pas.

```
int n = …;
int f = 1;
while (n >= 2) {
      f = f * n;
      n = n - 1;
}
```

3. Traduisez ce programme à l'aide des instructions du processeur. On suppose que la variable n est rangée en mémoire à l'adresse 1000_{10} et que f est rangée à l'adresse 1002_{10}. Comme en TD, vous supposerez pour simplifier que le programme est chargé en mémoire à l'adresse 0 et que chaque instruction du processeur a une taille de 1 octet.

```
00 : LDA    #1      { int f = 1 ; }
01 : STA    1002

02 : LDA    1000    { while (n >= 2) ; CMP #2 effectue n – 2 : on sort si n – 2 < 0 }
03 : CMP    #2
04 : BRN    12      { adresse de « STOP » }

05 : LDA    1002    { f = f * n }
06 : MUL    1000
07 : STA    1002

08 : LDA    1000    { n = n – 1 }
09 : SUB    #1
10 : STA    1000

11 : BRA    3       { adresse de « CMP #2 » }
12 : STOP
```

Note : sachant que n est déjà dans D0, f = f * n peut aussi s'écrire :

```
MUL   1002
STA   1002
```

Annexe

Jeu d'instructions du processeur

LDA : charger l'accumulateur
STA : stocker l'accumulateur
ADD : ajouter à l'accumulateur
SUB : soustraire de l'accumulateur
MUL : multiplier l'accumulateur
DIV : diviser l'accumulateur
CMP : comparer à l'accumulateur
BRx : se brancher si x (x = Z, N, C, ou V) est positionné
BRA : se brancher systématiquement
STOP : arrêter le processeur