A = 5

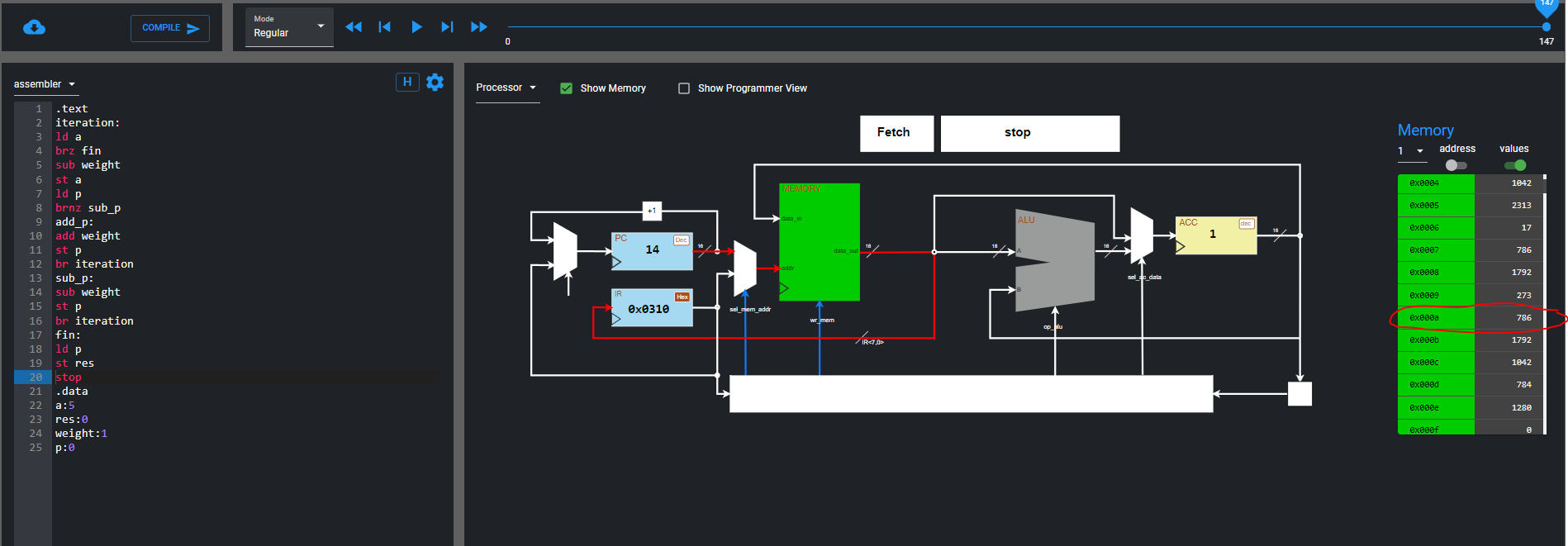
Partie 1 :

**Q1) Quelles sont les structures de contrôle utilisées au sein de ce programme ?**

Les structures de contrôle utilisées dans ce programme sont : *BRZ, BRNZ, BR*

**Q2/0,25 point Quel est le contenu en mémoire à l’adresse 0x0010 (qui diffère selon votre valeur de data) à la fin de l’exécution de ce programme ?**

Le contenu en mémoire a l’adresse *0x0010 est 786* a la fin de l’execution de ce programme.



**Q3/0,25 point Que fait ce programme ? Répondez à la question en décrivant le principe de fonctionnement du programme en évitant de référer à son contenu ligne par ligne.**

Le programme boucle tant que la valeur de a n’est pas nulle en sachant que celle-ci est décrémenté à chaque boucle. Le reste (res) est égal a la valeur de p nous informe sur le caractère pair ou impair du nombre du boucle effectué. Par exemple, si la valeur de a est impaire, le reste prendre la valeur 1.

**Q4/0,25 point Si le processeur est sur 32 bits et n’utilise pas le bit le plus significatif pour représenter le signe, quel serait le résultat maximal du programme ? Justifier votre réponse.**

Pour 32 bits : 4294967295

Partie 2

Avec l’éditeur d’assembleur de Code Machine pour le processeur à accumulateur sans registre MA, écrivez un programme qui permet de trouver le maximum de deux valeurs disjointes, de même signe, a et b. Dans le cas positif, vous pouvez assumer que a et b se situent toujours entre 1 et 99. Dans le cas négatif, vous pouvez assumer que a et b se situent toujours entre -99 et -1. Le maximum doit être stocké dans la variable res dans la section data à la fin de l’exécution du programme. Par exemple, pour a=-10 et b=-13, le programme retournera res=-10. Pour a=10 et b=13, le programme retournera res=13. Le choix de a et de b sont à votre discrétion, mais nous devrions pouvoir les changer aisément (toujours en respectant les critères mentionnés ci-haut) pour obtenir un nouveau maximum (0,10 point pour le fonctionnement statique du programme et 0,40 point pour son extensibilité). Discutez de votre approche et des limites qui s’imposent sur le choix du processeur (0,25 point).

Ce code sert dans un premier temps à comparer deux nombres positifs compris entre [1, 99] pour déterminer lequel est le maximal. Pour ce faire, on décrémente de 1 chacun des nombres jusqu’à un des deux atteint le zéro et par conséquent, ce nombre est le plus petit et on stocke le nombre max dans la variable res.

Si ces deux nombres sont négatifs la variable compteur va incrémenter jusqu’à 0 pour ensuite reload les valeurs des deux nombres et les comparer en les traitant comme deux nombres négatifs.

**Q0/0,00 point Indiquez votre valeur de data dans votre rapport.**

**Q1/0,15 point Quelles sont les structures de contrôle utilisées au sein de ce programme ?**

Les structures de contrôle utilisées dans ce programme sont : *BRZ, BRNZ, BR*

**Q2/0,10 point Quel est le contenu en mémoire à l’adresse 0x002A (qui diffère selon votre valeur de data) à la fin de l’exécution de ce programme ?**

À la fin de l’exécution du programme le contenu en mémoire à l’adresse 0x002A est 0.

**Q3/0,25 point Que fait ce programme ? Répondez à la question en décrivant le principe de fonctionnement du programme en évitant de référer à son contenu ligne par ligne. Indice : Pour quel(s) valeur(s) de data ne trouve-t-on pas un 1 à l’adresse 0x002A ?**

Le boucle le nombre de fois égale à la valeur stocker dans data. Et pour toutes les valeurs != 1 stocker dans le programme, la valeur à l’adresse 0x002A = 0. Donc se programme détecte si la valeur stocker dans data est de 1, si c’est le cas le programme retourne 1 sinon, le programme retourne 0.

If (data == 1) {

Return 1 (true).

}

Else return 0 (false)

**Q4/1 point Écrivez votre version en langage C du programme. Ce dernier devrait compiler et s’exécuter sans erreur, et ce, avec le même comportement et le même résultat que le code en assembleur fourni (0,5 point). Votre programme devrait inclure des commentaires, un échange (printf) avec l’utilisateur et une gestion intelligente de la mémoire qui devrait être libérée à la fin du programme (aucune fuite de mémoire) (0,25 point). Enfin, votre programme se doit d’être facilement extensible et modifiable (notamment en ce qui concerne la variable data) (0,25 point).**

3.2 Partie 2 Q1/0,75 point À l’aide du jeu d’instructions étendu du processeur à accumulateur avec registre MA, écrivez un programme assembleur qui permet de calculer le 7 + abs((MATRICULE\_1 + MATRICULE\_2) % 13) terme de la suite de Fibonacci. Le programme doit être le plus court possible et il ne doit seulement garder en mémoire que les trois derniers termes calculés (valeur courante de la suite, celle précédente et celle qui précède la précédente) (0,25 point). À noter que 0,50 point est accordé à cette question si le code fourni peut être facilement modifié pour calculer le dernier M terme voulu de la suite de Fibonacci, et ce, en changeant simplement la valeur d’une variable dans la partie .data du programme. Attention au plagiat.