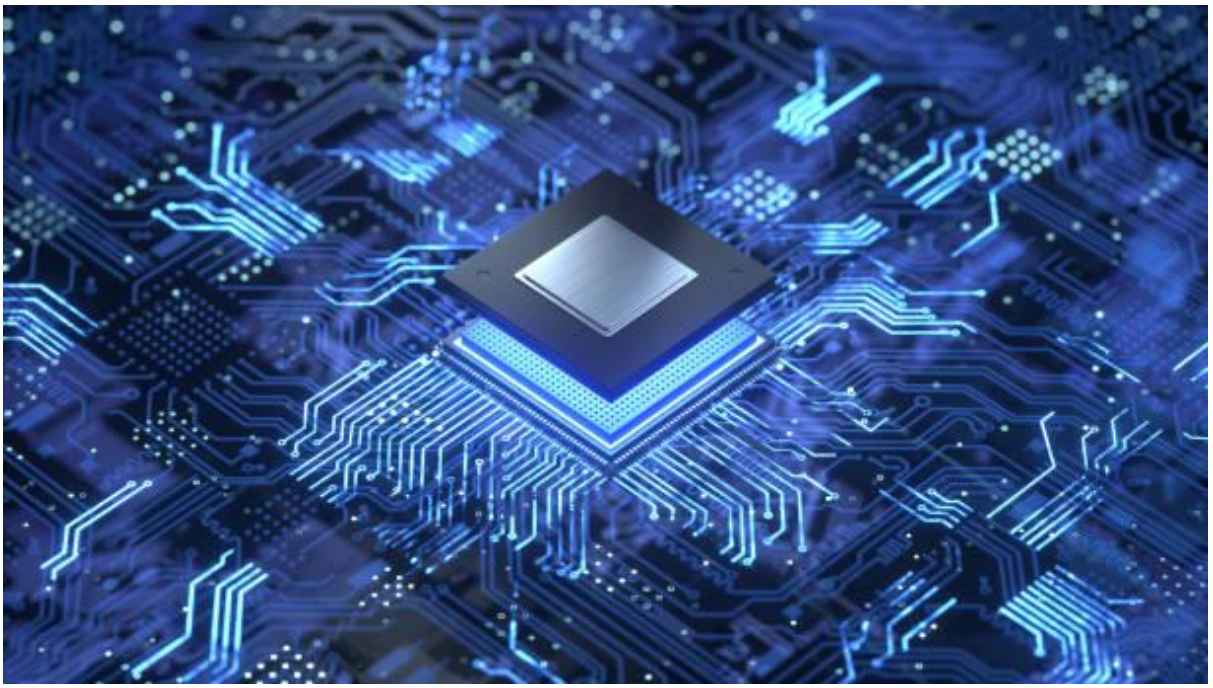
Rendu Mini-Projet R204

Programmation en Langage Assembleur



2B	Charpentier Alexandre	Baco Alistair

Introduction

Voici notre rendu pour le projet R204 à faire en autonomie.
Vous trouverez ci-dessous une explication des champs et des instructions que nous avons créés ainsi que les codes des exercices demandés et leurs explications.

Les Champs

Nom	Taille	Adresse de début	Valeur par défaut	Explication
JMP	1	1	0	Active un bus qui fait sauter la mémoire à une instruction donnée inconditionnellement
JMPZ	1	2	0	Active un bus qui fait sauter la mémoire à une instruction donnée si le flag Z est a 1
JMPNZ	1	3	0	Active un bus qui fait sauter la mémoire à une instruction donnée si le flag Z est a 0
JMPN	1	4	0	Active un bus qui fait sauter la mémoire à une instruction donnée si le flag N est à 1
JMPPZ	1	5	0	Active un bus qui fait sauter la mémoire à une instruction donnée si le flag N est à 0
MUX1	2	6	00	Active le MUX1 qui selon la valeur indiquée sur 2 bits choisira d'insérer dans l'ALU à l'opérande I1, la valeur des registres B, SI, DI ou de la RAM
MUX0	1	8	0	Active le MUX0 qui selon la valeur indiquée sur 1 bits choisira d'insérer dans l'ALU à l'opérande I0 la valeur du

				registre A ou la valeur indiquée dans le champs DATA
MUX2	1	9	0	Active le MUX2 qui selon la valeur indiquée sur 1 bits choisira de d'écrire dans la RAM à l'adresse contenu dans le répertoire SI ou DI
DEC	3	10	000	Active le décodeur qui selon la valeur indiquée sur 3 bits permet d'activer l'écriture les registres, A, B, SI, DI ou la RAM
ALU	4	13	0000	Active l'ALU avec différents calculs possibles choisis par le code indiqué
DATA	16	17	0000000000000000	Permet, si choisis par le MUX0, d'écrire une valeur sur 16 bits dans un registre ou de servir d'opérande à l'ALU dans I0.

Les Instructions

Nom	**Instructions**	**Valeur**	**Explication**
LOADA #valeur	MUX0 DEC ALU	0 001 0001	Écris dans le registre A une valeur donnée
LOADSI #valeur	MUX0 ALU DEC	0 0001 011	Écris dans le registre SI une valeur donnée
LOADADI	MUX1 ALU DEC	10 0010 001	Copie dans le registre A, le contenu du registre DI
LOADAADRSI	MUX1 DEC ALU	11 001 0010	Charge dans le registre A, la donnée de la RAM des données dont l'adresse se trouve dans le registre SI
LOADBADRDI	MUX1	11	Charge dans le registre B, la donnée de la

	ALU DEC MUX2	0010 010 1	RAM des données dont l'adresse se trouve dans le registre DI
LOADDIADRSI	MUX1 ALU DEC	11 0010 100	Charge dans le registre DI, la donnée de la RAM des données dont l'adresse se trouve dans le registre SI
INCSI	MUX1 ALU DEC	01 0101 011	Incrémente SI de 1
DECDI	MUX1 ALU DEC	10 0110 100	Décrémente DI de 1
JMP <label>	JMP	1	Effectue un saut inconditionnel à l'instruction étiquetée par <label>
JMPNZ <label>	JMPNZ	1	Effectue un saut à l'instruction étiquetée par <label> si le résultat de l'instruction précédente n'est pas nul
JMPPZ <label>	JMPPZ	1	Effectue un saut à l'instruction étiquetée par <label> si le résultat de l'instruction précédente n'est pas négatif (positif ou nul)
LOADASI	MUX1 ALU DEC	01 0010 001	Copie dans le registre A, le contenu du registre SI
LOADBADRSI	DEC MUX1 ALU	010 11 0010	Charge dans le registre B, la donnée de la RAM des données dont l'adresse se trouve dans le registre SI
LOADADRSIB	DEC	101	Copie le contenu du registre B, dans la RAM des données à l'adresse indiquée par le registre SI
CMPDIA	MUX1 MUX0 ALU	10 1 1011	Compare DI et A et met à jour les indicateurs en conséquence (l'UAL fait en interne la soustraction DI-A pour mettre à jour les indicateurs)

CMPB #valeur	ALU	1011	Compare B et la valeur immédiate qui suit et met à jour les indicateurs en conséquence (l'UAL fait en interne la soustraction B-valeur pour mettre à jour les indicateurs)
SUBB #valeur	ALU DEC	1000 010	Effectue la soustraction B – valeur et stocke le résultat dans B
JMPN <label>	JMPN	1	Effectue un saut à l'instruction étiquetée par <label> si le résultat de l'instruction précédente est négatif
CMPSIA	MUX1 MUX0 ALU	01 1 1011	Compare SI et A et met à jour les indicateurs en conséquence (l'UAL fait en interne la soustraction SI-A pour mettre à jour les indicateurs)
CMPBA	MUX0 AMU	1 1011	Compare B et A et met à jour les indicateurs en conséquence (l'UAL fait en interne la soustraction B-A pour mettre à jour les indicateurs)

Question 2 - Programme Palindromes

```
01              LOADSI #0
02              LOADDIADRSI
03      main    INCSI
04              LOADBADRDI
05              LOADAADRSI
06              CMPBA
07              JMPNZ npalin
08              LOADADI
09              CMPSIA
10              JMPPZ palin
11              DECDI
12              JMP main
13      npalin  LOADA #-1
14              JMP fin
15      palin   LOADA #1
16      fin     JMP fin
```

Ligne 1 : Instanciation du registre SI qui nous servira de compteur au fur et à mesure du programme

Ligne 2 : récupération de la longueur du mot dans le registre DI (valeur dans la RAM à l'adresse indiquée par SI)

Ligne 3 : début de la partie principale du programme, on commence par augmenter la valeur dans le registre SI de 1, ce qui a la première itération va décaler l'adresse correspondante dans la RAM a la première lettre et non à la longueur du mot. Dans le reste du programme, cette incrémentation servira simplement à passer à la lettre suivante.

Ligne 4 : On met dans B la lettre du mot à l'adresse dans la RAM indiquée par la valeur dans DI (à la première exécution cela sera la dernière lettre du mot)

Ligne 5 : On met dans A la lettre du mot à l'adresse dans la RAM indiquée par la valeur dans SI (à la première exécution cela sera la première lettre du mot)

Ligne 6 : on compare B et A, si leur code ascii est le même le résultat de la comparaison doit être nul, signifiant que le mot est pour l'instant un palindrome. Si le résultat n'est pas nul cela signifie que le mot n'est pas un pas un palindrome

Ligne 7 : Si le résultat de la comparaison n'est pas nul cela signifie que le mot n'est pas un palindrome, on jump donc à la fin correspondant au fait que le mot ne soit pas un palindrome
Ligne 8 : Nous avons fini le traitement de deux lettres du mot, il faut maintenant vérifier que nous n'avons pas fini de traiter le mot, c'est ce que nous allons faire jusqu'à la ligne 12
Nous commençons par mettre la valeur de DI dans A, cela est fait pour pouvoir comparer SI et DI puisque selon les instructions données nous ne pouvons pas directement comparer SI et DI

Ligne 9 : Ici Nous effectuons la comparaison entre SI et DI, correspondant respectivement aux index des lettres d'un côté et de l'autre du mot, DI correspond à la partie droite du mot et doit par conséquent avoir une valeur plus grande que SI, si c'est le cas le programme continue, sinon…

Ligne 10: Si la valeur de SI est supérieure ou égale à celle de DI, cela signifie que toutes les lettres du mot on déja été comparées entre elles et que le mot est effectivement un palindrome, on jump ainsi à la fin

Ligne 11: Si nous sommes arrivés à cette ligne, nous savons que pour l'instant le mot est un palindrome mais que nous n'avons pas fini de le traiter, nous allons donc faire en sorte de continuer le traitement, on décale DI d'une lettre vers la gauche en le décrémentant

Ligne 12 : Toute potentielle condition a été traitée précédemment, nous pouvons ainsi jump au début de la fonction principale qui va décaler SI d'une lettre vers la droite pour continuer le traitement

Lignes 13 & 14 : Si nous arrivons à ces lignes, c'est que le mot n'est pas un palindrome nous faisons alors le nécessaire pour le signaler à l'utilisateur, nous mettons -1 dans le registre A puis le programme se fini en déclenchant un jump vers l'instruction finale

Ligne 15 : SI nous arrivons à cette ligne c'est que le mot est un palindrome on le signale alors à l'utilisateur en mettant 1 dans le registre A, ici il n'y a pas besoin de jump vers la fin puis la fin se situe à la ligne suivante

Ligne 16 : Cette ligne déclenche un jump vers elle-même pour éviter que l'exécution continue dans une autre partie de la rom, qui n'a rien à voir avec le programme.

Question 4 - Programme MAJUSCULE

```
01              LOADSI #0
02              LOADDIADRSI
03  pasmin INCSI
04              LOADASI
05              CMPDIA
06              JMPN fin
07              LOADBADRSI
08              CMPB #97
09              JMPN pasmin
10              CMPB #123
11              JMPPZ pasmin
12              SUBB #32
13              LOADADRSIB
14              JMP pasmin
15      fin     JMP fin
```

Ligne 1 : Comme dans le premier programme, au début de la RAM se situe la longueur du mot traité, nous instancions SI à 0

Ligne 2 : Nous mettons dans DI la longueur du mot qui nous permettra au fur et à mesure de l'exécution du programme de vérifier que nous ne sommes pas à la fin du mot, à la manière d'une boucle for dans un langage de haut niveau

Ligne 3 : à cette ligne est le départ du programme principal, qui commence par le passage à la lettre suivante pour SI

Lignes 4 & 5 : On met dans A la valeur de SI pour comparer SI à DI puisque d'après les limitations d'instructions nous ne pouvons pas comparer directement SI à DI. Ensuite nous comparons SI et DI.

Ligne 6 : Si le résultat de la comparaison est négatif cela signifie que la valeur dans SI est supérieure à la valeur dans DI, et que donc le mot est fini, le programme déclenche donc un jump vers la fin

Ligne 7 : On met dans B la valeur à l'adresse actuelle de SI (à la première itération de la boucle le code ascii de la première lettre, deuxième itération la deuxième etc) dans le but de changer ou non cette lettre.

Ligne 8 : On compare le code ascii de la lettre avec celui de la première lettre minuscule du code ascii, en effet notre programme ne doit faire l'opération de mettre en majuscule que sur les caractères qui sont des lettres minuscules, au risque de changer le caractère affiché.

Ligne 9 : Si le résultat de la comparaison précédente est négatif, cela signifie que la valeur dans le registre B correspond à un caractère dont le code ascii est avant celui du 'a' minuscule, un caractère qui n'est donc pas à traiter, le programme déclenche alors un jump qui va permettre de traiter le caractère suivant.

Ligne 10 : En suivant le même principe qu'à la ligne 8, nous allons cette fois comparer B avec la valeur 123, un de plus que la valeur de la dernière lettre du code ascii.

Ligne 11 : arrivés à cette vérification, puisque les tests précédents ont été passés nous savons que la valeur du code ascii traité est supérieure ou égale à 97. Ce test permet de vérifier qu'elle soit inférieure à 123, ce qui nous garantit d'être dans l'ensemble des lettres minuscules. C'est pour cela que le programme va déclencher un passage au traitement d'une autre lettre uniquement si le résultat est positif ou nul, signifiant que la lettre n'est pas une lettre minuscule.

Ligne 12 : une fois arrivés à cette ligne, nous sommes sûrs que le code ascii de la lettre soit compris entre 97 et 122 (97 et 122 inclus) nous pouvons donc mettre la lettre en majuscule en soustrayant 32 à son code ascii (qui correspond à l'écart entre une lettre majuscule et sa minuscule correspondante).

Lignes 13 & 14 : On remplace dans la RAM le nouvelle valeur de la lettre puis on jump au passage à une lettre suivante.
Ligne 15 : Cette ligne déclenche un jump vers elle-même pour éviter que l'exécution continue dans une autre partie de la rom, qui n'a rien à voir avec le programme. Cette ligne n'est accessible que par un jump, lorsque la valeur de SI est supérieure à celle de DI et que donc le traitement du mot est fini.