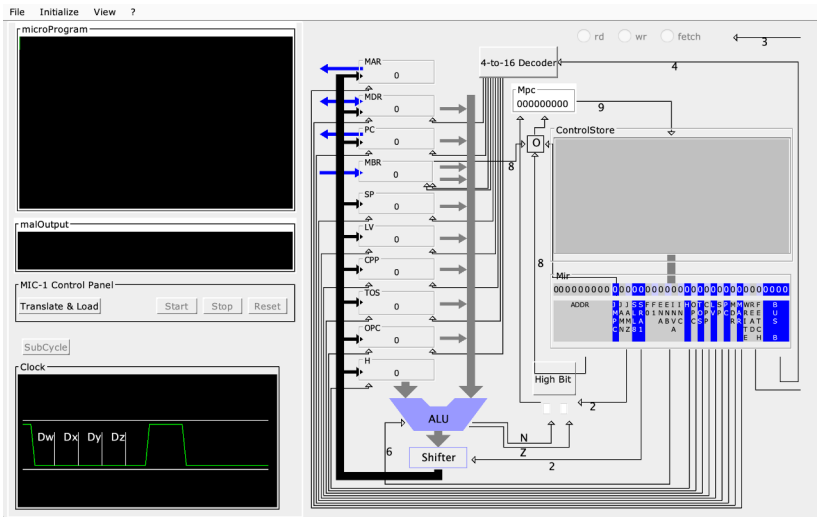
Dans cette feuille d'exercice, vous utiliserez emuMIC, le simulateur de la micro-architecture MIC1. Pour sauver votre code le plus simple est de faire des copier/coller.

Exercice 1.Lancez le simulateur emuMic qui est dans Célène. Vous obtenez cet affichage :



a. Repérez :
   i. la zone pour le code,
   ii. la zone où s'affiche le microcode binaire
   iii. la machine IJVM :
      - registres
      - ALU
      - Bus
      - décodeur
      - registres de contrôle :
        1. MPC
        2. MIR
      - flag de retour d'état : N et Z
   iv. Flèches figurant l'accès à la mémoire.

b. Tester ce microcode élémentaire : `main1 : H=1`

c. Que signifie cet affichage ? Expliquez-le en détail.

```
0 |00000000000000110001100000000001111
```

d. Exécuter le code « sous-cycle par sous-cyle ». Comment évolue le fonctionnement de la machine ?

e. Comment la machine boucle-t-elle ?

Exercice 2.Proposez un microprogramme qui permet de:

a. mettre la valeur 1 dans les registre OPC, TOS

b. et 0 dans MAR et MDR.

c. Poursuivez en ajoutant à la fin la microinstruction de l'exercice 1 (pour cela positionnez le label `main1` sur la première ligne). Observez le champ *Adresse Suivante*

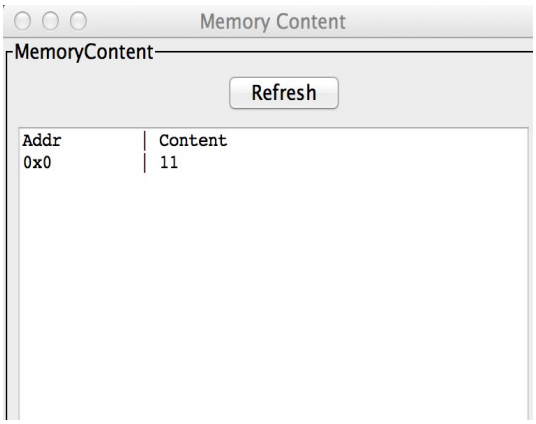
Exercice 3.Concevez un microprogramme qui permet d'affecter ainsi les registres:

a. OPC=1, TOS=-1 , MAR=0 et MDR=1

b. OPC=2, TOS=4 , MAR=6 et MDR=8

Exercice 4.Dans la mémoire de votre simulateur positionnez le mot de 32 bits 8 à l'adresse de mot:1 via un microprogramme

Exercice 5.Avec l'interface du simulateur (menu « Initiatize/Memory ») affectez la valeur 11 à l'adresse 0. Écrivez ensuite un programme qui lit cette valeur et la met dans TOS.

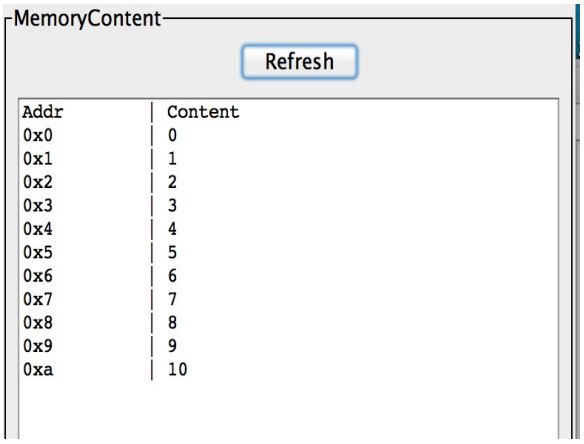


Exercice 6. Ecrire un microcode qui additionne 2 valeurs consécutives (par exemple 11 et 8) en mémoire (adresses 0x00 et 0x01) et place en mémoire le résultat à l'adresse 0x02.

Exercice 7.Placez la valeur x=`252313600` à l'adresse mémoire 0x00.

a. Puis lisez l'adresse 0x00 en utilisant MAR/MDR/rd. Que récupère-t-on comme valeur ?
b. Recommencez avec l'adresse 0x01 en utilisant MAR/MDR/rd. Que récupère-t-on comme valeur ?
c. Puis lisez à nouveau l'adresse 0x01 mais cette fois en utilisant PC/MBR/Fetch. Placez le résultat dans OPC. Que ce passe-t-il ? Expliquez. (Vous pouvez observer la valeur de x en hexadécimal pour mieux comprendre ce qui c'est passé)
d. Conclusion sur le rôle de PC/MBR/Fetch ? Pourquoi se comporte-t-il différemment ? A quoi sert-il ?

Exercice 8.Ecrivez un microcode qui lit le mot ***n*** à l'adresse 0 (par exemple 9) et qui rempli les ***n*** (par exemple 9) premières adresses mémoires avec les nombres allant de 1 à ***n***. **Exemple ci-contre:**



Exercice 9.Écrivez ce microprogramme qui effectue un if :

```
MDR=0
boucle: N=0 ; if (N) goto fin; else goto deb
deb:TOS=0 ; goto fin
TOS=1
fin:H=0
```

```
POS|MIC-INSTRUCTION
0  |10000000000000010000010000000001111
1  |00000001100000010000000000100001111
2  |10000000000000110001001000000001111
3  |00000000010000100000000000000001111
256|00000000100000010000100000000001111
```

a. Observez le binaire généré du microcode et repérez chacune des lignes de code.
b. Expliquez l'ordre des microinstructions.
c. Faite tourner le code avec MDR=0 en première ligne puis avec MDR=1. Observez la différence et expliquez comment le microcode effectue un goto conditionnel.

Exercice 10.Testez les exemples du TD et en particulier les cas interdits par le compilateur MAL.