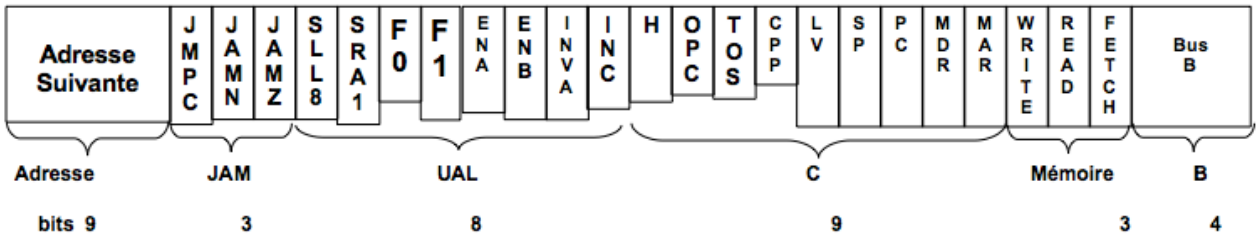
Dans les premiers exercices de cette feuille, vous utiliserez emuMIC, le simulateur de la micro-architecture MIC1. Pour sauver votre code le plus simple est de faire des copier/coller.

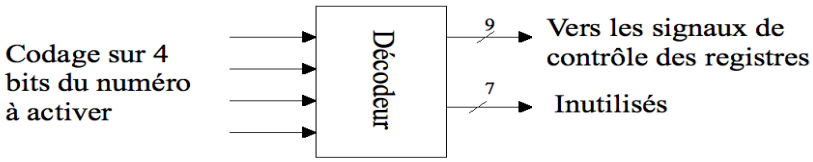
Exercice 1.Voici une microinstruction écite en MAL (Micro Architecture Language):

```
TOS=MAR=OPC+H;rd;goto main1
```

Donnez son équivalent en binaire sur 36 bits.

Rappelons que comme un seul des 9 registres peut être écrit sur B à un instant, 4 bits suffisent pour coder le registre concerné en utilisant un décodeur 1 parmi 16 dont 7 possibilités ne sont pas utilisées. Il suffit de 4 bits au lieu de 9. C'est précisé ci-dessous.





Registres sur B

0 = MDR	**5 = LV**
1 = PC	**6 = CPP**
2 = MBR	**7 = TOS**
3 = MBRU	**8 = OPC**
4 = SP	**9-15 rien**

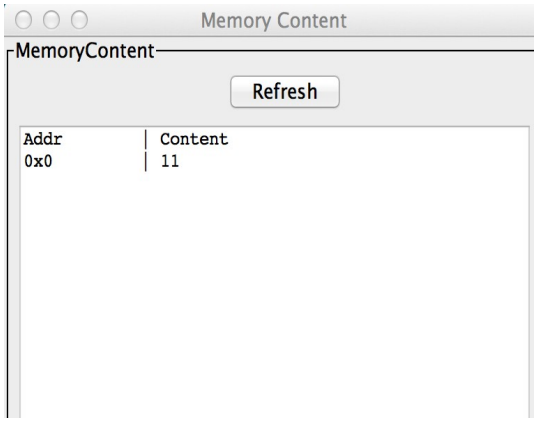
On a alors 29-9+4=24 signaux pour contrôler le chemin de

Exercice 2.Concevez 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 champs *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:*1* via un microprogramme

Exercice 5.Avec l'interface du simulateur 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.Modifier le microprogramme pour ajouter une instruction IJVM ISA.

Dans cet exercice, on étend les capacités de Mic1 en modifiant sa mémoire de commande pour y ajouter une instruction. Nous allons ajouter le détail de cette commande dans les fichiers de définition de l'assembleur IJVM puis écrire un programme IJVM pour utiliser cette nouvelle instruction.

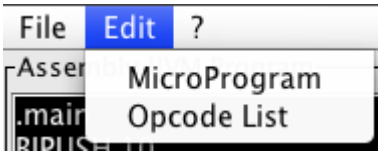
Nous allons ajouter l'instruction `COM`, qui signifie complément. Cette instruction dépile un mot, calcule le complément à un de ce mot et pousse le résultat sur la pile.

Le microcode est aisé mais il permet de comprendre l'ensemble du processus avant d'attaquer des exemples plus intéressants.

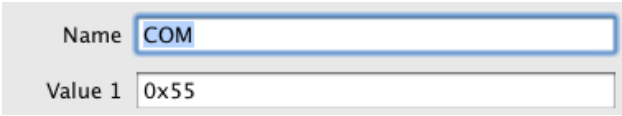
```
com1    MDR = TOS = not(TOS)   // calc 1s compl, save in TOS and MDR
com2    MAR = SP; wr; goto Main1  // set MAR from SP, write, recycle
```

Expliquez les deux microinstructions ici proposées.

Ces lignes doivent être incluses dans le microcode de l'émulateur EMUIJVM. Pour cela utiliser le menu edit/Opcode List.

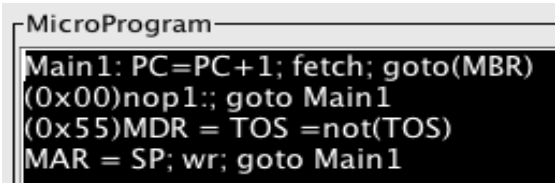


Avec le bouton NEW il est possible de déclarer une nouvelle instruction IJVM. Il faut lui donner un nom et un code hexadécimal.



Le code `0x55` est donc le code d'opération de COM. Il doit être spécifié aussi dans la partie Microprogramme.

A présent vous compilez la définition du microprogramme avec la commande Translate.



Pour tester votre nouvelle instruction, vous pouvez utiliser ce code là:

```
.main                   // start of program

.var                    // local variables for main program
.end-var
            BIPUSH 12
            DUP
            COM
            BIPUSH 5
            IAND
.end-main
```

a. Expliquez le comportement de ce programme.

b. Observez le déroulement du microprogramme et repérez vos instructions.

Exercice 8. Étendez le microcode d'IJVM pour inclure l'instruction DUPTWO qui met 2 fois le sommet de pile en sommet de pile.

Exercice 9. Ecrivez le microcode pour Mic1 qui implémente l'instruction IJVM POPTWO. Cette instruction supprime 2 mots sur le sommet de la pile.

Passez à l'exerciseur pour vous entrainer sur des cas plus pointus et intéressants.
`http://info.iut45.univ-orleans.fr/docs/ijvm/en/latest/tp2.html`