Guide de programmation du simulateur de processeur

Architecture

8 registres :

- R0 à R6 : registres de 32 bits
- SP : pointeur de pile

Mémoire : 1K mots de 32 bits.

Opérandes

- Entiers naturels sur 32 bits
- Entiers relatifs sur 32 bits en complément à 2
- Caractères en code ASCII
- Chaînes de caractères de codes ASCII
- Pointeurs constitués d'une adresse sur 10 bit

JEU D'INSTRUCTIONS

Instructions de déplacements de données

Instruction	Opération	Type d'opérandes
LD *dest*, *source*	*dest* ← *source*	dest : RG ou [RG] source : RG, [RG], [RG+d],Var,Val
ST *source*, *Var*	*Var* ← *source*	*source* : RG, [RG]
LEA *dest*, *Var*	*dest* ← adresse en mémoire de la variable désignée	dest :RG, [RG]
SWAP *oper*	échange les 16 bits de fort poids et les 16 bits de faible poids de oper	oper : RG, [RG], [RG+d], Var

Instructions arithmétiques

Instruction	Opération	Type d'opérandes
ADD *op1* , *op2*	*op1* ← *op1* + *op2*	op1 : RG, [RG] op2 : RG, [RG], [RG+d], Var, Val
SUB *op1* , *op2*	*op1* ← *op1* - *op2*	
MUL *op1* , *op2*	*op1* ← *op1* * *op2* Multiplication d'entiers relatifs	
MULU *op1* , *op2*	*op1* ← *op1* * *op2* Multiplication d'entiers naturels	
DIV *op1* , *op2*	*op1* ← *op1* / *op2* Division d'entiers relatifs	
DIVU *op1* , *op2*	*op1* ← *op1* / *op2* Division d'entiers naturels	
INC *oper*	*oper* ← *oper* + 1	oper : RG, [RG], [RG+d], Var
DEC *oper*	*oper* ← *oper* - 1	
NEG *oper*	*oper* ← - (*oper*)	

Remarque : les instructions arithmétiques, à l'exception de NEG, positionnent les indicateurs de débordement des entiers naturels et relatifs du registre d'état de l'unité de traitement.

Instructions logiques

Instruction	Opération	Type d'opérandes
OR *op1* , *op2*	*op1* ← *op1* OU *op2*	op1: RG, [RG] op2 : RG, [RG], [RG+d], Var,Val
AND *op1* , *op2*	*op1* ← *op1* ET *op2*	
NOT *oper*	*oper* ← COMPLEMENT *oper*	oper : RG, [RG], [RG+d], Var

Instruction de comparaison

Instruction	Opération	Type d'opérandes
CMP *op1* , *op2*	compare *op1* à *op2, positionne le registre d'état*	op1: RG, [RG] op2 : RG, [RG], [RG+d], Var,Val

Instructions de décalage

DEBER est l'indicateur de débordement des entiers naturels du registre d'état de l'unité de traitement.

Instruction	Opération	Type d'opérandes
SHR *oper*	décalage logique à droite	oper : RG, [RG], [RG+d], Var
SHL *oper*	décalage logique à gauche	
SAR *oper*	décalage arithmétique à droite	
SAL *oper*	décalage arithmétique à gauche	
ROR *oper*	décalage cyclique à droite	
ROL *oper*	décalage cyclique à gauche	

Instructions de branchement (rupture de séquence)

Instruction	Opération	Type d'opérandes
JMP *etiq*	branchement inconditionnel	*etiq* : étiquette (instruction où aller).
Bxx *etiq*	branchement conditionnel	

condition de branchement	comparaison d'entiers naturels	comparaison d'entiers relatifs
égal	BEQ	BEQ
différent	BNE	BNE
inférieur	BLTU	BLT
inférieur ou égal	BLEU	BLE
supérieur	BGTU	BGT
supérieur ou égal	BGEU	BGE
débordement	BDEBU	BDEB

Instructions relatives à la pile

Instruction	Opération	Type d'opérandes
PUSH *oper*	empile *oper*. (SP ← SP-1 puis Mem(SP) ← *oper*)	oper : RG, [RG], [RG+d], Var
PULL *oper*	dépile *oper*. (*oper* ← Mem(SP) puis SP ← SP+1)	

Instructions d'appel et retour de sous programmes

Instruction	Opération	Type d'opérandes
CALL *etiq*	appel du sous-programme référencé par *etiq*	*etiq* : étiquette
RET	retour de sous-programme	
RET Val	retour de sous-programme avec vidage de la pile	Val : val

Instruction de contrôle

Instruction	Opération	Type d'opérandes
HLT	arrêt du processeur	

Instructions relatives à l'unité d'échange (Entrées / Sorties)

Instruction	Opération	Type d'opérandes
IN *oper*, *port*	*oper* ← octet du registre de l'UE désigné par son n°.	*Oper :* RG, [RG] *Por t :* entier naturel, n° de port de l'UE. (sur 8 bits de faible poids *)*.
OUT *oper*, *port*	octet du registre de l'UE désigné par son n° ← *oper*.	

L'ASSEMBLEUR DU SIMULATEUR

Déclaration des variables :

Instruction	Opération	Type d'opérandes
nom **DSW** *taille*	Déclaration sans initialisation	Nom facultatif (réservation de place)
nom **DW** *valeur*	Déclaration avec initialisation	

Squelette d'un programme en assembleur :

```
        .DATA
                définition des variables et constantes
        .CODE
        LEA     SP,STACK
        ..... suite du code
        dernière instruction (en général HLT)
truc:   ..... corps de la procédure truc (1ere procédure)
        ret     x
        ..... autres procédures
machin: ..... corps de la procédure machin (dernière procédure)
        RET     x
        .STACK          taille
```

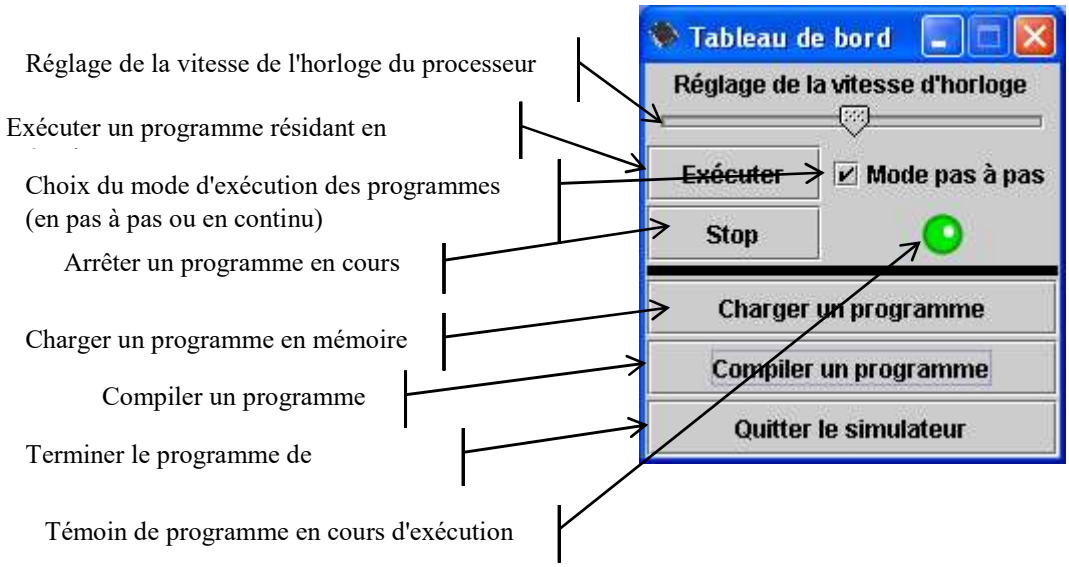
UTILISATION DU SIMULATEUR POUR COMPILER ET EXECUTER UN PROGRAMME

Présentation :

Fenêtres du simulateur :

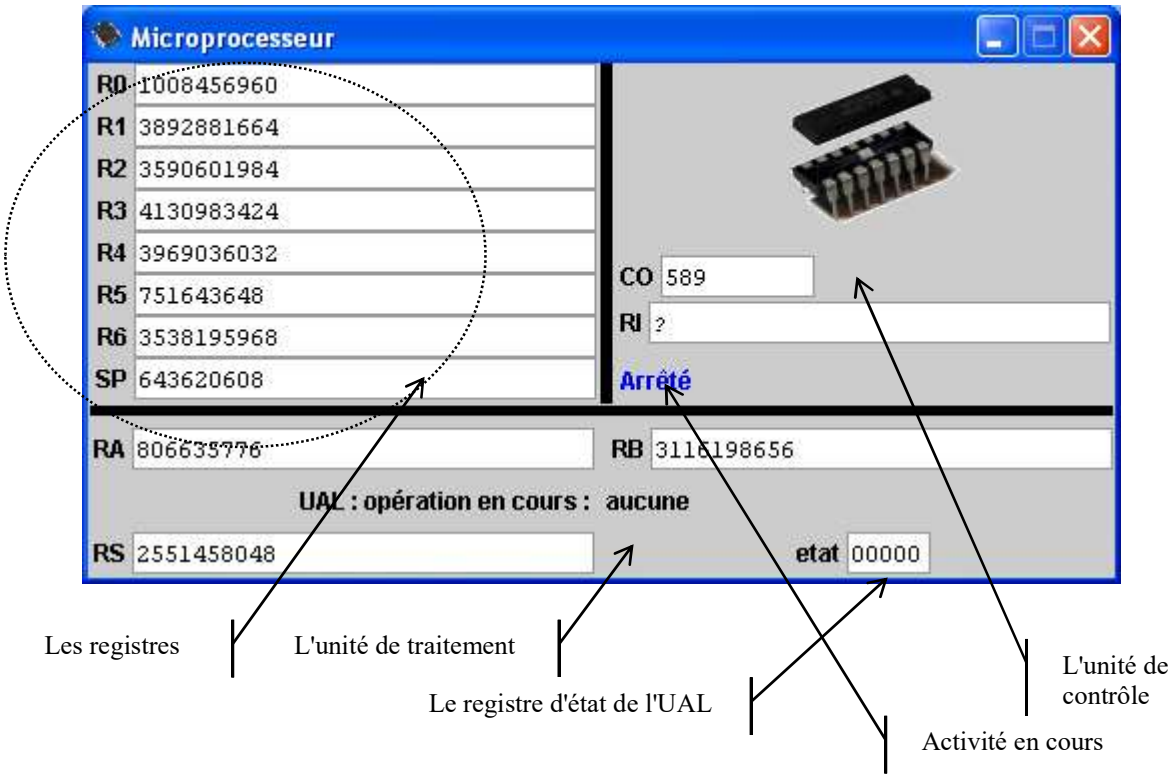
- Le panneau de contrôle qui permet de compiler des programmes et des exécuter.
- Le processeur qui contient les registres, l'unité de traitement et l'unité de contrôle.
- La mémoire.
- L'unité d'échange.
- Un périphérique constitué d'un écran graphique de 256x256 et d'un clavier à 6 touches.

Le Panneau de contrôle



Le Processeur :

Le processeur possède 6 registres d'usage général (R0 à R6) plus un registre de sommet de pile SP.



L'unité de contrôle contient :

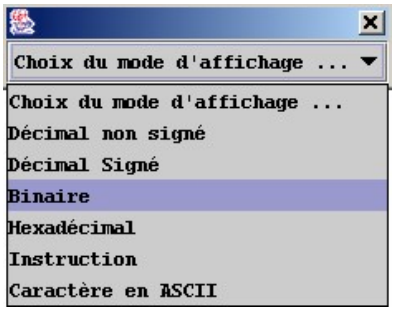
- Le compteur ordinal CO
- Le registre d'instructions RI
- Une ligne d'état indiquant l'activité en cours : recherche d'instruction, décodage d'instruction, exécution d'instruction ou arrêt.

L'unité de traitement contient une UAL avec ses registres d'entrée (RA et RB), de sortie (RS) et son registre d'état.

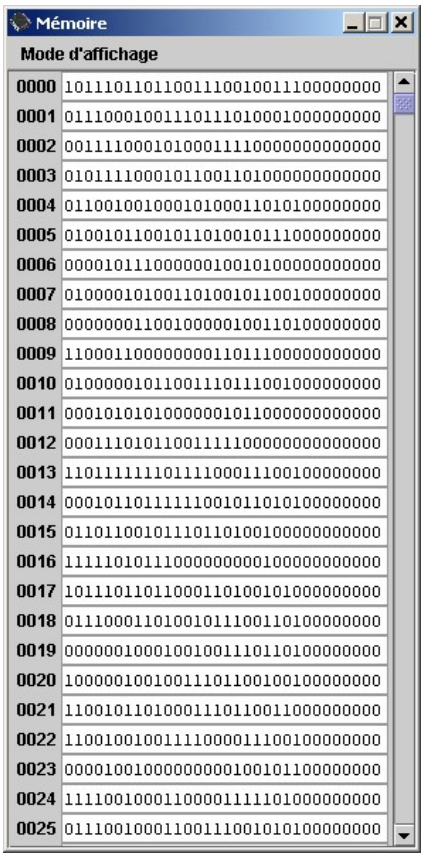
Le registre d'état contient 5 bits. Il a la structure suivante :

Indicateur d'égalité	Indicateur d'infériorité entre entiers naturels	Indicateur d'infériorité entre entiers relatifs	Indicateur de débordement d'opération entre entiers naturels	Indicateur de débordement d'opération entre entiers relatifs

- L'indicateur d'égalité est mis à 1 par une instruction de comparaison (CMP) lorsque les 2 opérandes sont égaux.
- L'indicateur d'infériorité entre entiers naturels est mis à 1 par une instruction de comparaison (CMP) lorsque le 1er opérande considéré comme un EN est inférieur au second opérande considéré lui aussi comme un EN.
- L'indicateur d'infériorité entre entiers relatifs est mis à 1 par une instruction de comparaison (CMP) lorsque le 1er opérande considéré comme un ER est inférieur au second opérande considéré lui aussi comme un ER.
- L'indicateur de débordement d'opération entre entiers naturels est mis à 1 par une instruction arithmétique lorsque le résultat considéré comme un EN déborde.
- L'indicateur de débordement d'opération entre entiers relatifs est mis à 1 par une instruction arithmétique lorsque le résultat considéré comme un ER déborde.



Remarque : lorsque l'on clique sur un registre, une fenêtre de dialogue permettant de choisir le mode selon lequel le contenu de ce registre sera affiché à l'écran (voir ci-contre) apparaît. Bien entendu cela ne modifie pas le contenu du registre mais seulement son mode d'affichage.



La Mémoire :

Elle est constituée de 1024 mots de 32 bits.

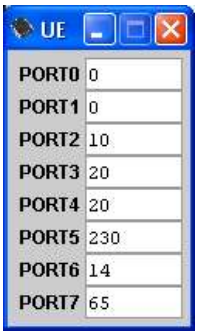
Un ascenseur permet de faire défiler l'ensemble de son contenu.

Remarque : lorsque l'on clique sur un mot de la mémoire une fenêtre de dialogue permettant de choisir le mode selon lequel le contenu de ce mot sera affiché à l'écran (voir ci-dessus) apparaît. Bien entendu cela ne modifie pas le contenu du mot mémoire mais seulement son mode d'affichage.

La ligne de menu "Mode d'affichage" permet d'appliquer un mode d'affichage à une zone de mémoire comprise entre deux adresses. Lorsque l'on clique sur ce menu, on voit apparaître une fenêtre permettant de choisir l'adresse de début et celle de fin de la zone de mémoire à laquelle va s'appliquer le mode d'affichage choisi.

Afin d'en faciliter la lecture, lorsqu'un programme est chargé dans la mémoire le mode d'affichage de chaque mot est adapté selon le type de son contenu. Il peut être modifié par la suite en cliquant dessus.

L'unité d'échange :



Elle contient 8 registres de 8 bits correspondant aux numéros de port 0 à 7.

Elle permet de connaître l'état des touches et de dessiner ou écrire dans la fenêtre graphique du périphérique. Elle permet aussi de détecter les mouvements et les clics de la souris dans la zone graphique de l'écran du périphérique.

Son fonctionnement est le suivant :

1°) ENTREES

Le clavier

Les touches du clavier positionnent des valeurs dans le Port0 selon le principe suivant :

Etat du clavier		*Numéro de la touche*					
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Etat du clavier indique les actions qui ont eu lieu sur le clavier depuis la dernière fois que le Port0 a été lu :

- 00 rien ne s'est produit
- 11 une touche a été appuyée
- 10 une touche a été lâchée

Le bit 7 permet donc de savoir si un événement s'est produit, tandis que le bit 6 précise lequel.

Remarque : Ces deux bits sont automatiquement remis à 0 lorsque le processeur lit le contenu du Port0.

Numéro de la touche indique le numéro de la touche appuyée ou lâchée. Les numéros des touches sont les suivants :

- 1 pour la touche ↑ soit 00 0001
- 2 pour la touche → soit 00 0010
- 3 pour la touche ↓ soit 00 0011
- 4 pour la touche ← soit 00 0100
- 5 pour la touche **A** soit 00 0101
- 6 pour la touche **B** soit 00 0110

Remarque : Ces 6 bits sont automatiquement remis à 0 lorsque le processeur lit le contenu du Port0.

L'écran

- Lorsque la souris survole l'écran, ses coordonnées sont écrites dans Port6 (coordonnée en x) et Port7 (coordonnée en y). Un programme peut donc en permanence savoir où se trouve la souris sur l'écran en lisant ces 2 ports de l'UE.
- Lorsque l'on clique avec la souris sur l'écran, ses coordonnées sont écrites dans Port6 (coordonnée en x) et Port7 (coordonnée en y) et le Port 0 est positionné de la même façon que si l'on avait appuyé une touche de numéro 7 (valeur : 11000111).

Remarque : Le Port0 est automatiquement remis à 0 lorsque le processeur en lit le contenu.

2°) SORTIES

Les registres Port1 à Port5 pilotent l'écran graphique. Le registre Port5 indique l'opération graphique à exécuter tandis que les registres Port1 à Port4 en constituent les paramètres.

Dès qu'une valeur est écrite dans Port5, la commande est exécutée en utilisant les registres Port1 à Port4 comme paramètres (il faut donc les avoir préparés avant).

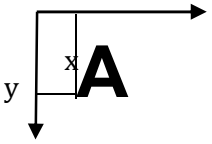
Le Port5 est divisé en 2 demi-registres de 4 bits chacun, les 4 bits de gauche indiquent la couleur dans laquelle doit se faire le tracé (si la commande est une commande de dessin ou d'écriture). Les 4 bits de droite indiquent la commande à exécuter :

Couleur de tracé				*Commande graphique*			
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Les *Commandes graphiques* sont les suivantes :

- 0000 : effacer l'écran, les 4 bits *Couleur de tracé* sont ignorés.
- 0001 : tracer le point dont les coordonnées sont dans Port1 (coordonnée en x) et Port2 (coordonnée en y). La couleur de tracé est celle indiqué par les 4 bits *Couleur de tracé.*
- 0010 : tracer la ligne dont les coordonnées du point de départ sont dans Port1 (coordonnée en x) et Port2 (coordonnée en y) celles du point d'arrivée sont dans Port3 (coordonnée en x) et Port4 (coordonnée en y). La couleur de tracé est celle indiqué par les 4 bits *Couleur de tracé.*
- 0011 : tracer le rectangle dont les coordonnées du coin supérieur gauche sont dans Port1 (coordonnée en x) et Port2 (coordonnée en y), la largeur est dans Port3 et la hauteur dans Port4. La couleur de tracé est celle indiqué par les 4 bits *Couleur de tracé.*
- 0100 : tracer l'ellipse inscrite dans le rectangle dont les coordonnées du coin supérieur gauche sont dans Port1 (coordonnée en x) et Port2 (coordonnée en y), la largeur est dans Port3 et la hauteur dans Port4. La couleur de tracé est celle indiqué par les 4 bits *Couleur de tracé.*
- 0101 : tracer le rectangle plein dont les coordonnées du coin supérieur gauche sont dans Port1 (coordonnée en x) et Port2 (coordonnée en y), la largeur est dans Port3 et la hauteur dans Port4. La couleur de remplissage est celle indiqué par les 4 bits *Couleur de tracé.*
- 0110 : tracer l'ellipse pleine inscrit dans le rectangle dont les coordonnées du coin supérieur gauche sont dans Port1 (coordonnée en x) et Port2 (coordonnée en y), la largeur est dans Port3 et la hauteur dans Port4. La couleur de remplissage est celle indiqué par les 4 bits *Couleur de tracé.*
- 0111 : écrire aux coordonnées placées dans Port1 (coordonnée en x) et Port2 (coordonnée en y) le caractère dont le code ASCII est placé dans Port3. La couleur d'écriture est celle indiqué par les 4 bits *Couleur de tracé.* Les valeurs des coordonnées en x (Port1) et en y (Port2) sont automatiquement mises à jour pour pouvoir écrire le caractère suivant sur la même ligne ou, si elle est pleine, sur la ligne suivante.

Les coordonnées en x et y du caractère correspondent au point en bas à gauche de ce caractère

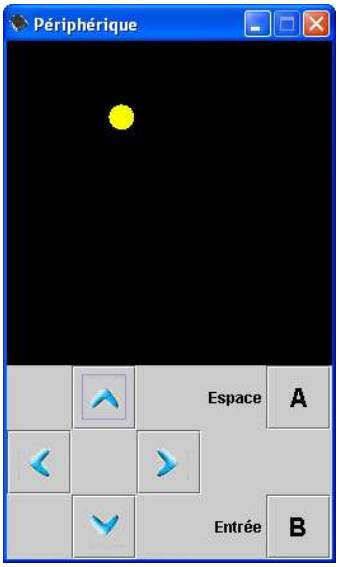


Les *Couleurs de tracé* sont les suivantes :

0	0000 : noir	4	0100 : bleu foncé	8	1000 : vert clair	C	1100 : rouge vif
1	0001 : gris foncé	5	0101 : bleu clair	9	1001 : magenta	D	1101 : rouge clair
2	0010 : gris moyen	6	0110 : cyan	A	1010 : orange	E	1110 : jaune
3	0011 : gris clair	7	0111 : vert	B	1011 : rose	F	1111 : blanc

Le périphérique :

1°) L'écran



L'écran est graphique d'une définition de 256 pixels par 256 pixels. Les coordonnées (0,0) correspondent au coin supérieur gauche. Les coordonnées en x (horizontalement) augmentent quand on se déplace vers la droite, les coordonnées en y (verticalement) augmentent quand on se déplace vers le bas.

L'écran détecte les mouvements et les clics de la souris sur sa surface.

2°) Les touches

Le périphérique comporte 6 touches (comme une gameboy simple) : 4 touches de direction et 2 touches de commande (A et B). Les touches peuvent être activées à la souris ou en utilisant le clavier.

Pour pouvoir utiliser le clavier la fenêtre du périphérique doit être la fenêtre active. Les touches correspondantes du clavier sont les flèches pour les touches de direction, la touche Espace pour la touche de commande A et la touche Entrée pour la touche de commande B.

Utiliser le simulateur :

Le simulateur est contenu dans un fichier d'archives Java (**simulateur.jar**). Le programme peut être lancé en double cliquant sur le fichier jar ou par la commande : **java –jar simulateur.jar**

Pour exécuter un programme :
1°) Taper le programme source à l'aide d'un éditeur de textes quelconque en mode txt et l'enregistrer sous un nom de la forme : ***xxxx.asm***

2°) Appeler le compilateur par le bouton approprié du simulateur. Après avoir choisi un fichier .asm à compiler, une fenêtre s'ouvre pour indiquer l'état de la compilation (et les erreurs éventuelles). Si la compilation ne génère pas d'erreurs, un fichier exécutable xxxx.mpc est obtenu et placé dans le même répertoire que xxxx.asm. On peut alors fermer la fenêtre du compilateur.

Remarque : Au moment de la compilation le réglage de vitesse d'horloge et l'état de la case à cocher "pas à pas" du simulateur sont mémorisés. Ainsi, lorsque le fichier sera chargé en mémoire, la case à cocher et l'horloge retrouveront les valeurs qu'elles avaient au moment de la compilation du fichier.

3°) Charger le programme ainsi obtenu par le bouton approprié du simulateur qui permet de choisir le fichier .mpc à exécuter. Le programme est maintenant visible en mémoire et le CO est initialisé sur la première instruction. En outre, la case à cocher et l'horloge ont retrouvé les valeurs qu'elles avaient au moment de la compilation de ce fichier. Elles peuvent, bien évidemment, être modifiées.

4°) Lancer le programme par le bouton Exécuter. Le curseur du haut permet de régler la vitesse de l'horloge du processeur. Ainsi les instructions s'exécuteront plus ou moins vite selon ce réglage. Ceci permet en particulier de suivre l'exécution d'un programme et de vérifier que tout se passe bien.

Chaque fois qu'un registre est modifié il apparaît un bref instant en rouge. Chaque fois qu'un registre est lu il apparaît un bref instant en vert. Lorsque l'horloge est mise à sa vitesse maximale, les changements de couleurs des registres ne sont plus visibles.

La case à cocher permet de choisir un fonctionnement en pas à pas ou en continu. En mode pas à pas le processeur s'arrête à la fin de chaque instruction. Pour qu'il effectue l'instruction suivante du programme il faut réactiver le bouton Exécuter. En mode continu il exécute les instructions l'une après l'autre jusqu'à rencontrer une instruction HLT. On peut toutefois l'arrêter en utilisant le bouton Stop. Toutefois ce bouton n'arrête l'exécution qu'après que l'instruction en cours soit terminée.