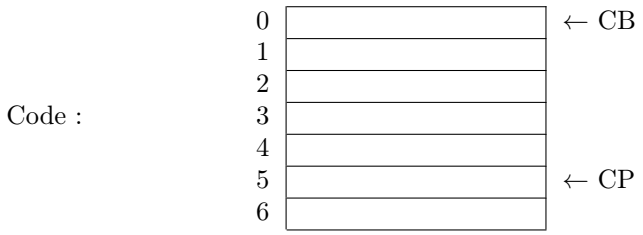
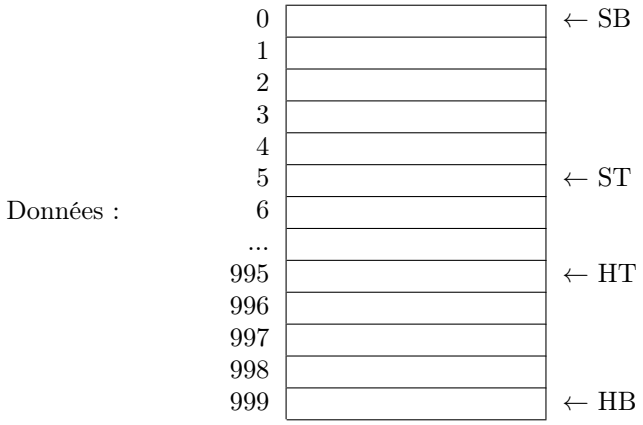
1 La machine TAM

Machine à pile. Pas de registre de donnée.





Instructions (16) dont :

PUSH n	ST = ST+n
POP (d) n	a = ST -d; ST = ST - d -n; Pour i de d à 0 Donnees(ST++) = Donnees[a++] fin pour
LOADL n	Donnees(ST)= n; ST = ST+1
LOADA d[r]	ST = d[r] ; ST = ST + 1;
LOAD (n) d[r]	Pour i de 0 a n-1 Donnees(ST+i) = Donnees(val(r)+d+i) fin pour; ST = ST+n
STORE (n) d[r]	Pour i de 0 a n-1 Donnees(val(r)+d+i) = Donnees(ST+i-n); fin pour; ST = ST-n
JUMP etiq	CP = val(etiq)
JUMP d[r]	CP = val(r) + d
JUMPIF (n) etiq	si Donnees(ST -1) = n alors CP = val(etiq) fin si; ST = ST -1
JUMPIF (n) d[r]	si Donnees(ST -1) = n alors CP = val(r) + d fin si; ST = ST -1
LOADI (n)	Empile n mots lus à l'adresse précedemment empilée
STOREI (n)	Ecrit les n mots empilés, à l'adresse empilée
SUBR op	Appel de op, consommation des arguments laissés en sommet de pile
HALT	Arret

Une étiquette pourra être déposée devant toute instruction comme dans l'exemple suivant :

```
; programme de test
debut:
  PUSH 1 ;@x
  JUMP main
fin:
  HALT
; affichage de x+10
main:
  LOADL 10;
  LOAD (1) 0[SB] ; x
  SUBR Iadd ; x + 10
etiq:
  SUBR Iout ; print x+10
  JUMP fin
```

2 Instructions de la machine TAM

Nom	Paramètres	Résultat	
Fonctions Booléens			
BNeg	1	1	Négation logique
BOr	2	1	Ou logique
BAnd	2	1	Et logique
BOut	1	0	Affiche sur `stdout` un booléen (`true` ou `false`)
BIn	0	1	Lit sur `stdin` un booléen (`true` ou `false`)
B2C	0	1	Conversion vers un caractère (`true` = `'1'`, `false` = `'0'`)
B2I	0	1	Conversion vers un entier (`true` = `1`, `false` = `0`)
B2S	0	1	Conversion vers une chaîne (`"true"`, `"false"`)
Fonctions Caractères			
COut	1	0	Affiche sur `stdout` un caractère
CIn	0	1	Lit sur `stdin` un caractère
C2B	1	1	Conversion vers un booléen (`'1'` = `true`, `'0'` = `false`)
C2I	1	1	Conversion vers un entier (le code ASCII)
C2S	1	1	Conversion vers la chaîne contenant seulement ce caractère
Fonctions Entiers			
INeg	1	1	Négation entière
IAdd	2	1	Addition entière
ISub	2	1	Soustraction entière
IMul	2	1	Multiplication entière
IDiv	2	1	Diviseur dans division entière
IMod	2	1	Reste dans division entière
IEq	2	1	Test égalité entre 2 entiers
INeq	2	1	Test différence entre 2 entiers
ILss	2	1	Test inférieur strictement entre 2 entiers
ILeq	2	1	Test inférieur ou égal entre 2 entiers
IGtr	2	1	Test supérieur strictement entre 2 entiers
IGeq	2	1	Test supérieur ou égal entre 2 entiers
IOut	1	0	Affiche sur `stdout` un entier
IIn	0	1	Lit sur `stdin` un entier
I2B	1	1	Conversion vers un booléen (1 = `true`, 0 = `false`)
I2C	1	1	Conversion vers un caractère (le code ASCII)
I2S	1	1	Conversion vers la chaîne représentant cet entier
Fonctions Mémoires			
MVoid	0	1	Renvoie la valeur « adresse non initialisée »
MAlloc	1	1	Alloue un bloc mémoire et renvoie son adresse
MFree	1	0	Libère un bloc mémoire
MCompare	2	1	Test égalité entre le contenu de 2 blocs mémoire
MCopy	2	0	Copie le contenu d'un bloc mémoire dans le second bloc mémoire
Fonctions Chaînes			
SAlloc	1	1	Création d'une nouvelle chaîne
SCopy	1	1	Création d'une copie de la chaîne passée en paramètre
SConcat	2	1	Création d'une nouvelle chaîne contenant la juxtaposition de deux paramètres
SOut	1	0	Affiche sur `stdout` une chaîne
SIn	0	1	Lit sur `stdin` une chaîne
S2B	1	1	Conversion vers un booléen (`"true"` = `true`, `"false"` = `false`)
S2C	1	1	Extraction du premier caractère de la chaîne
S2I	1	1	Conversion vers l'entier représenté par la chaîne