Jeu d'instructions du simulateur

ASR2-Système 2012-2013

20 mars 2013

Machine à mots de 16 bits, adresses sur 12 bits. Registres : accumulateur 16 bits, compteur de programme 12 bits.

Simulateur accessible sur `http://www.labri.fr/perso/billaud/WebSim16/` et `~/Bibliotheque/ASR2-systeme/WebSim16/index.html` au département.

1 Table des instructions

Code	Mnémonique	Description	Action	Cp =
0	`loadi` *imm12*	chargement immediat	`Acc = ext(imm12)`	`Cp + 1`
1	`load` *adr12*	chargement direct	`Acc = M[adr12]`	`Cp + 1`
2	`loadx` *adr12*	chargement indirect	`Acc = M[M[adr12]]`	`Cp + 1`
3	`store` *adr12*	rangement direct	`M[adr12] = Acc`	`Cp + 1`
4	`storex` *adr12*	rangement indirect	`M[M[adr12]] = Acc`	`Cp + 1`
5	`add` *adr12*	addition	`Acc += M[adr12]`	`Cp + 1`
6	`sub` *adr12*	soustraction	`Acc -= M[adr12]`	`Cp + 1`
7	`jmp` *adr12*	saut inconditionnel		`adr12`
8	`jneg` *adr12*	saut si négatif		`Acc<0 ? adr12 : Cp+1`
9	`jzero` *adr12*	saut si zero		`Acc==0 ? adr12 : Cp+1`
A	`jmpx` *adr12*	saut indirect		`M[adr12]`
B	`call` *adr12*	appel	`M[adr12] = Cp+1`	`M[adr12]+1`
C	`halt 0`	arrêt		
D		op. illégale	erreur	
E		op. illégale	erreur	
F		op. illégale	erreur	

Commentaires

- *adr12* (resp. *imm12*) désigne l'adresse (resp. la valeur immédiate) encodée sur les 12 bits de l'instruction
- l'instruction `loadi` procède à une *extension de signe* de *imm12* : le bit de poids fort de la valeur immédiate est copiée dans 4 bits de poids fort de l'accumulateur. Par exemple l'intruction `loadi -1` est codée `0000 1111 1111 1111` en binaire. Lors de l'affectation dans l'accumulateur 16 bits, le bit de signe de la valeur immédiate est propagé de façon à obtenir la valeur `1111 1111 1111 1111` (qui représente -1 sur 16 bits) dans l'accumulateur.
- lors de l'exécution des opérations indirectes (`loadx`, `storex`,`jmpx`) le contenu de `M[adr12]` qui est sur 16 bits est interprété comme une adresse sur 12 bits. Une erreur est détectée si les 4 bits de poids ne sont pas nuls, et entraîne l'arrêt du processeur.
- le paramètre de `halt` est ignoré.
- 3 codes ne sont pas utilisés.

2 Exemples de programme

```
#
# Calcul de la somme des entiers de 1 à N
#
    loadi 0     # S=0
    store S

    loadi 1     # K=1
    store K

BOUCLE          # si K>N aller à suite
    load  N     # - calcul N-K
    sub   K
    jneg  SUITE

    load  S     # S = S+K
    add   K
    store S

    loadi  1    # K = K+1
    add    K
    store  K
    jmp    BOUCLE

SUITE
    halt 0
#
# variables
#
N   word  5
K   word  0
S   word  0
```

```
# somme des éléments d'un tableau

   loadi 0
   store S
   store K
BOUCLE
   loadi 10        / loadi T
   sub   K        /  add   K
   jzero FIN     /   store PTR
   ..........   /    loadx PTR
   loadi 1       \   add   S
   add   K        \  store S
   store K
   jmp   BOUCLE
FIN
   halt  0
```