Utilisation de la pile pour passage de paramètres

Avant de faire un jsr, empiler les arguments à fournir au sous programme

Le sous programme peut dépiler

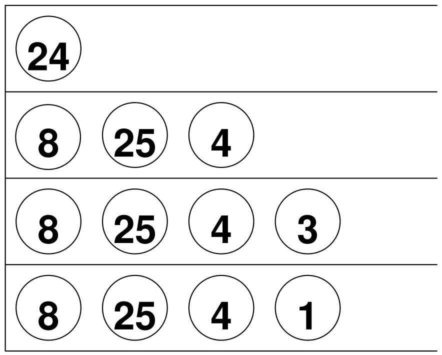
Nous avons donc besoin de deux instructions de plus:

- **push**
  - **empiler le contenu de R1**
- **pop**
  - **mettre le sommet de la pile dans R0 et dépilier**

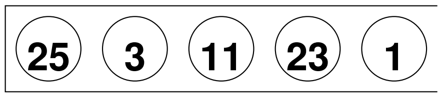
Ces deux instructions sont réalisables avec l'architecture actuelle

Implémentation de push et de pop

push:



pop:



Protocole d’appel de sous programmes

Un protocole d’appel est un ensemble de règles précises pour déterminer le partage de responsabilités entre un sous programme appelant et un sous programme appelé.

Exemple de protocole d’appel (plusieurs implémentations du langage C):

Appelant	Appelé
calculer argument n dans R1	
empiler R1	
...	
calculer argument 1 dans R1	
empiler R1	
jsr appelé	
	calcul éventuel de valeur dans R0
	ret
dépiler les n arguments	

t

Accès aux arguments dans un sous programme

Actuellement, la seule possibilité est d'utiliser pop, ce qui n'est pas pratique

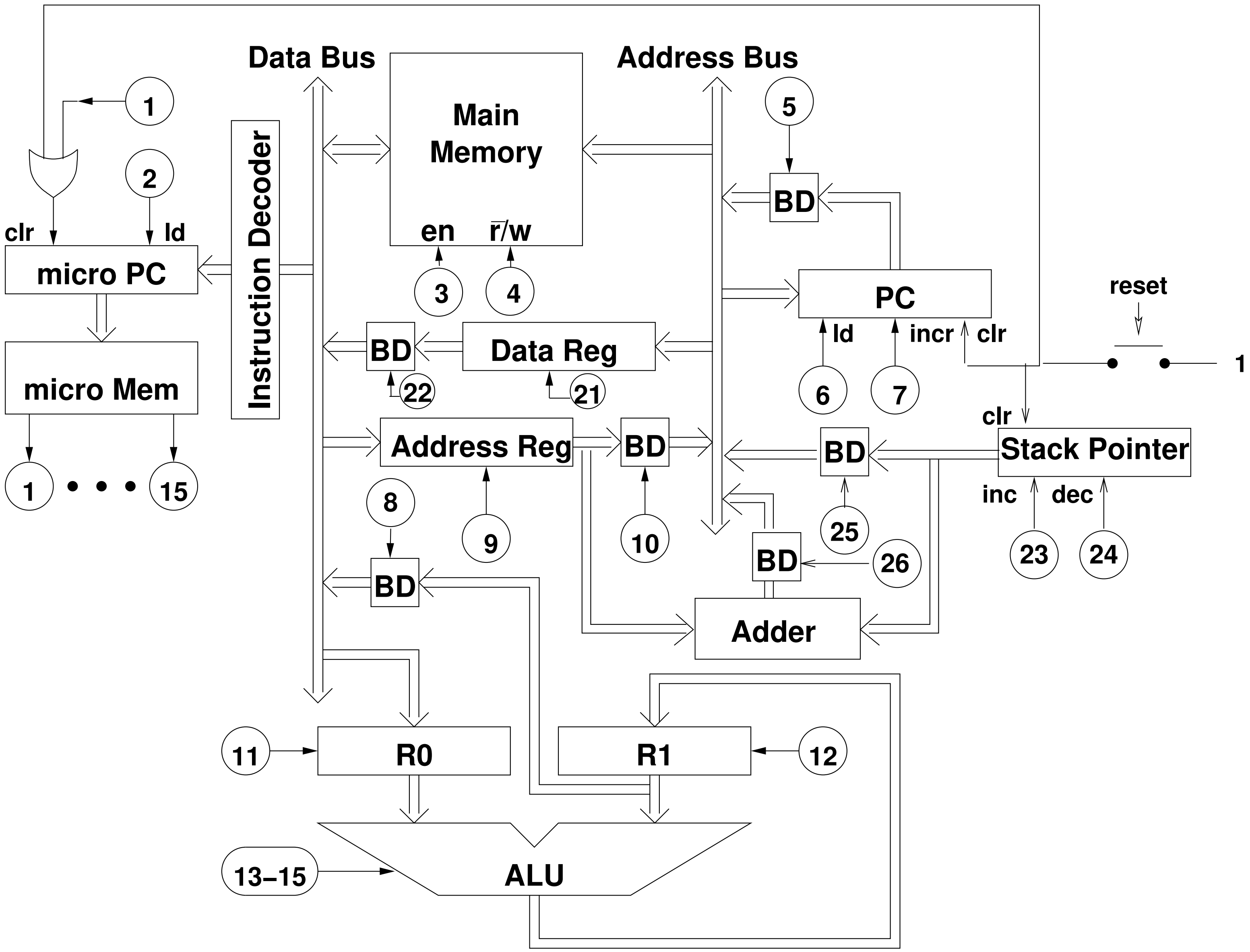
Il nous faut une possibilité d'accéder à une valeur en mémoire dont l'adresse est sp+constante

De cette manière, le sous programme peut récupérer argument i avec l'adresse sp+(i+1)

Nous avons donc besoin d'additionner le contenu de sp avec une constante.

Cette constante sera stockée après le code d'instruction, et copiée dans le registre d'adresses

Modifications pour l’accès aux arguments



Nouvelle instruction pour l'accès aux argument

Il est maintenant possible d'écrire une instruction lds (load from stack).

5	3	9	7
26	3	11	1

Dépiler les arguments

Après le retour de l'appelé, l'appelant doit dépiler les arguments

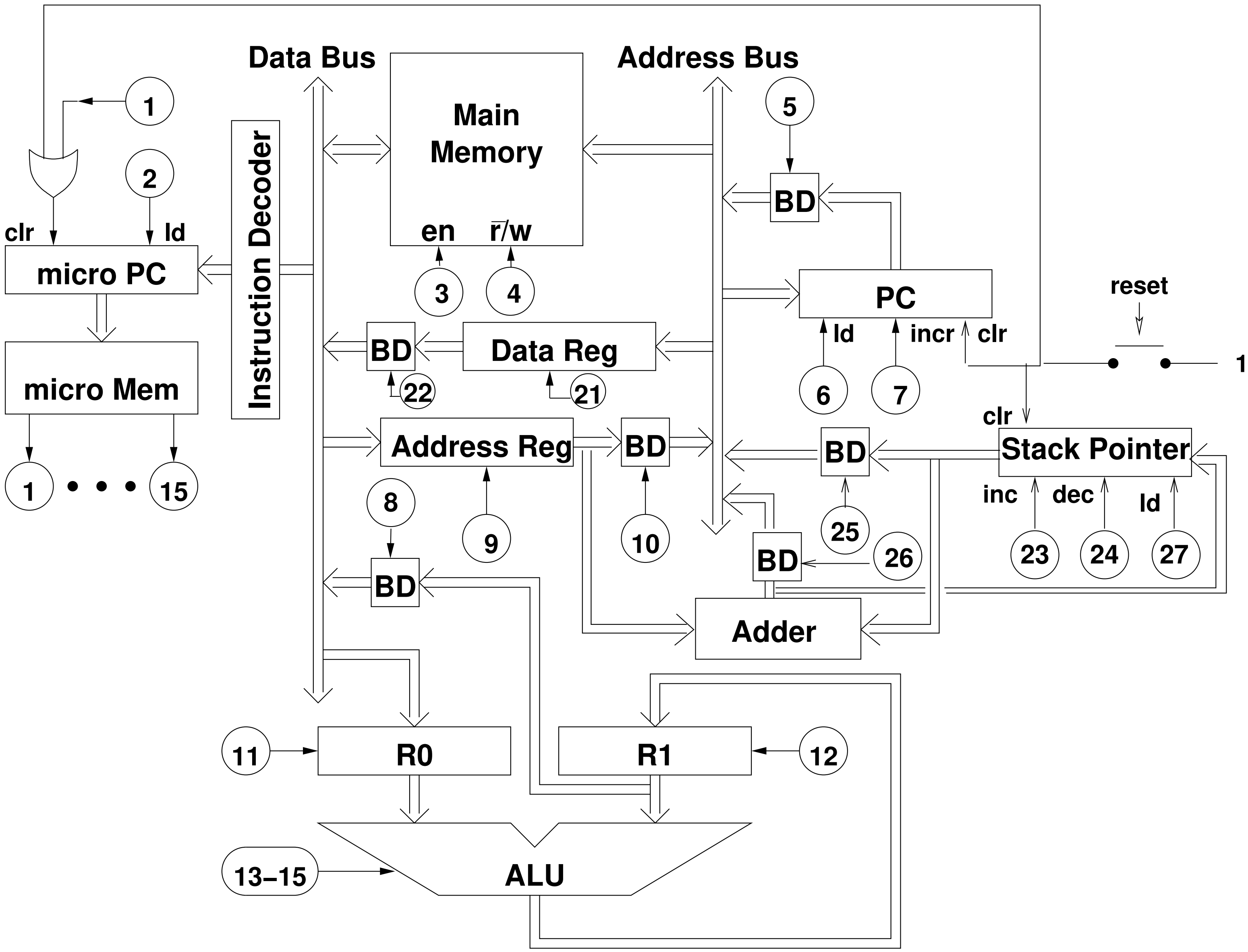
Actuellement, la seule possibilité est d'utiliser pop

Une amélioration possible est de pouvoir additioner une constante à sp

Il est possible d'utiliser l'additioneur existant, mais il faut pouvoir stocker le résultat dans sp

Ceci nécessite des fils supplémentaires, et une modification de sp pour permettre un signale ld

Modifications pour l'arithétique de sp



Nouvelle instruction pour l’arithmétique de sp

Il est maintenant possible d’écrire une instruction addsp

L’addition est toujours modulo 256, donc pas besoin de subsp

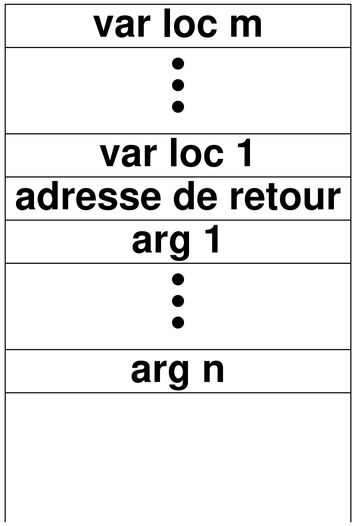
Voici l’implémentation

5	3	9	7
27	1		

Variables locales à un sous programme

Avec les instructions maintenant à notre disposition, il est possible d'avoir des variables locales allouées dans la pile

Voici le format de la pile pour un sous programme avec n arguments et m variables locales:



Nouveau protocole d'appel

Appelant	Appelé
calculer argument n	
empiler R1	
...	
calculer argument 1	
empiler	
jsr appelé	
	addsp –m
	calculer en utilisant lds, sts
	addsp m
	ret
addsp n	