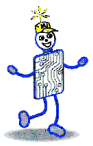
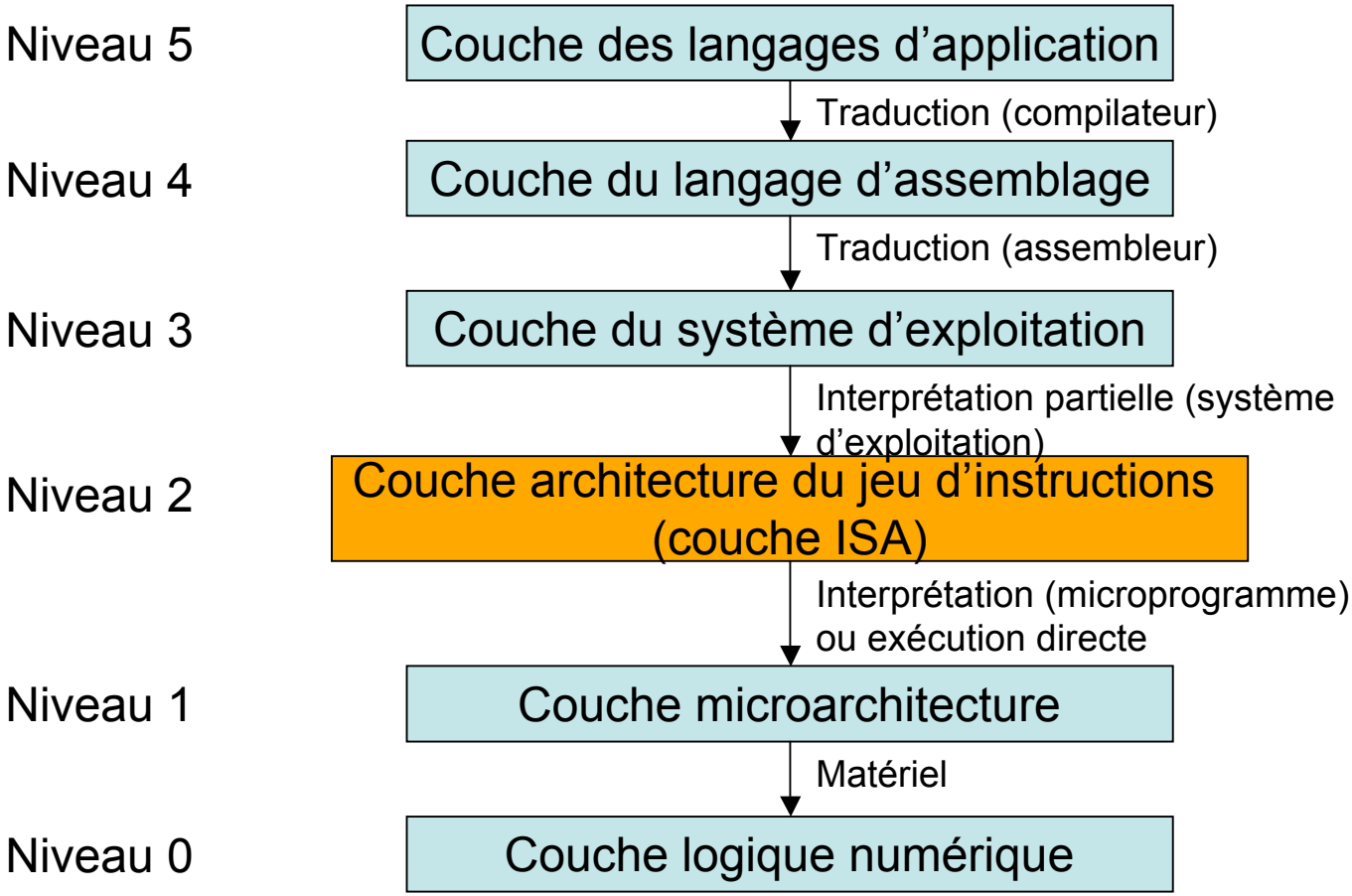
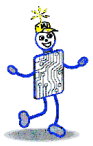


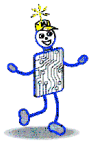
Jeu d'instructions





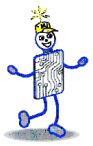
Instructions machines

- Les instructions et les données sont codées sur des mots mémoires (un ou plusieurs mots mémoires selon la nature de l'ordinateur)
- Les instructions machines sont propres à chaque microprocesseur
- Une instruction désigne un ordre donné au processeur et qui permet à celui-ci de réaliser un traitement élémentaire



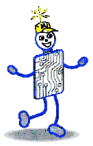
Jeu d'instructions

- Design définit les fonctions pouvant être exécutées par le processeur
- Différencie l'architecture de l'ordinateur
  - Nombre d'instructions
  - Complexité des opérations
  - Types de données supportés
  - Format
  - Utilisation de registres
  - Adressage (taille, modes)



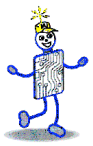
Classification des instructions

- Transfert de données (load, store)
  - Permettent de transférer une donnée depuis les registres du processeur vers la mémoire et vice versa ainsi qu'entre registres du processeur
  - Taille du *mot mémoire* ? 16? 32? 64 bits?
- Arithmétiques
  - Operateurs **+ - / * ^**
  - Entiers et en virgule flottante
- Logique Booléenne
  - **AND, XOR, NOT, …**
- Instructions manipulant un seul opérande
  - Négation, décrémentation, incrémentation, remise à 0

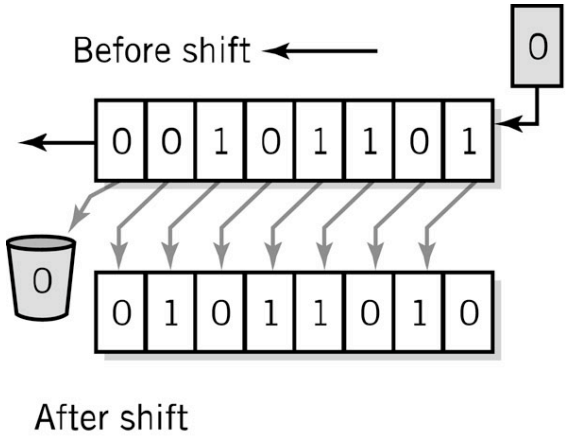


Classification des instructions

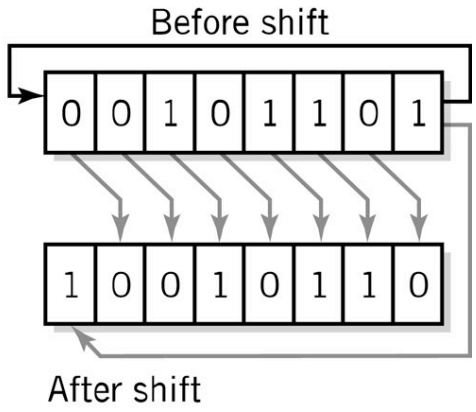
- Manipulation de bits
  - Flags pour tester les conditions
- Décalage/rotation
- Branchement ou de commande
- De la pile
- D’entrées/sorties
- Contrôle de la machine



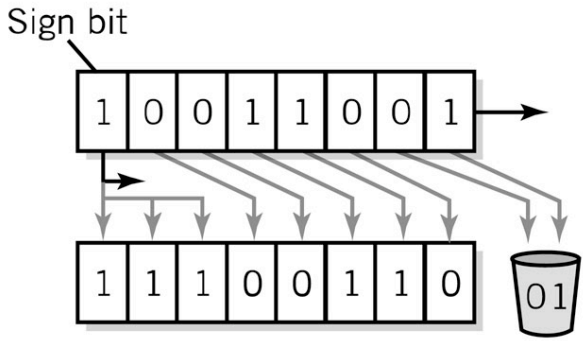
Décalage et rotation



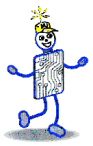
a. Left logical shift register 1 bit



b. Rotate right 1 bit

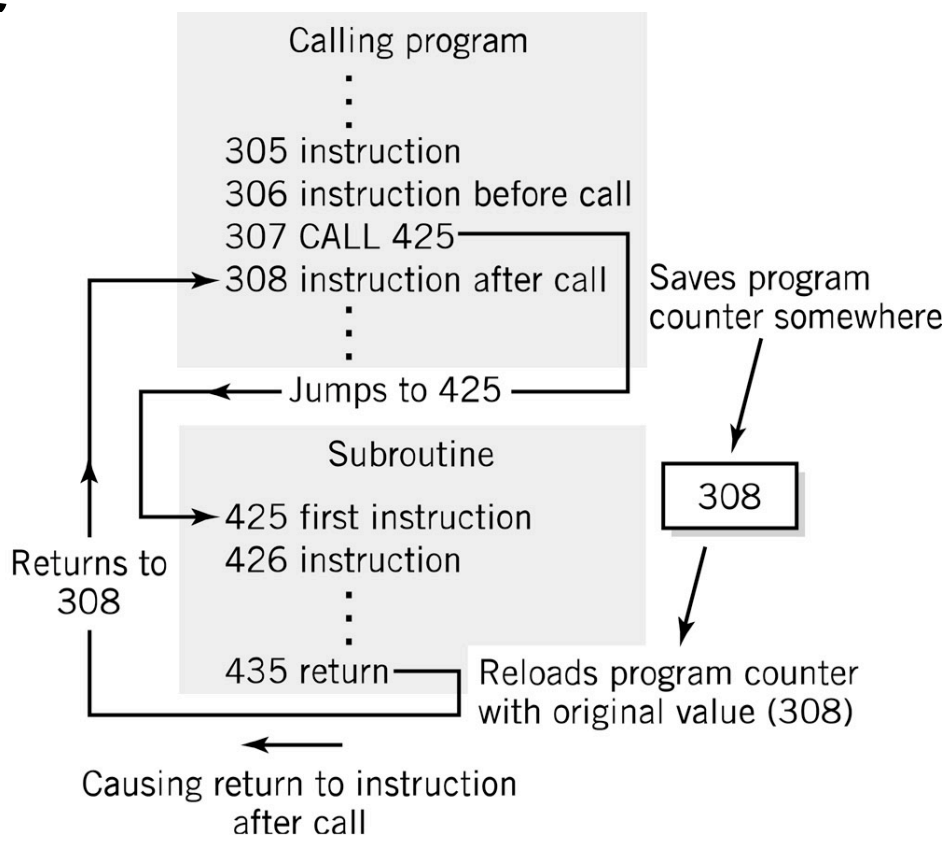


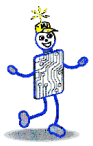
c. Right arithmetic shift 2 bits



Contrôle de programme

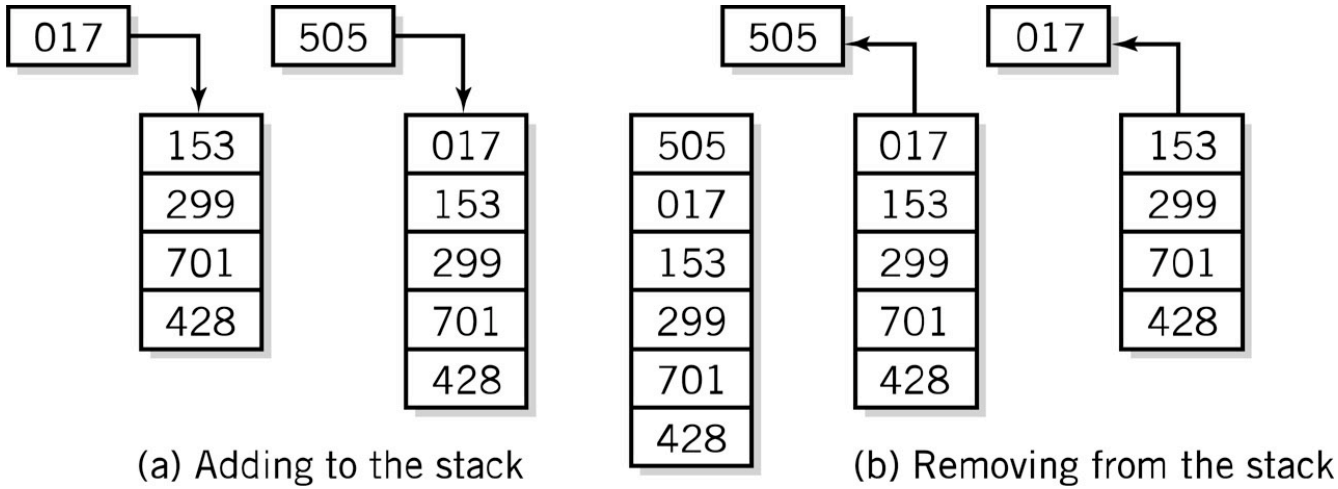
- Les instructions de saut ou de branchement
- Les instructions d'appels de sous-programmes





Instructions de la pile

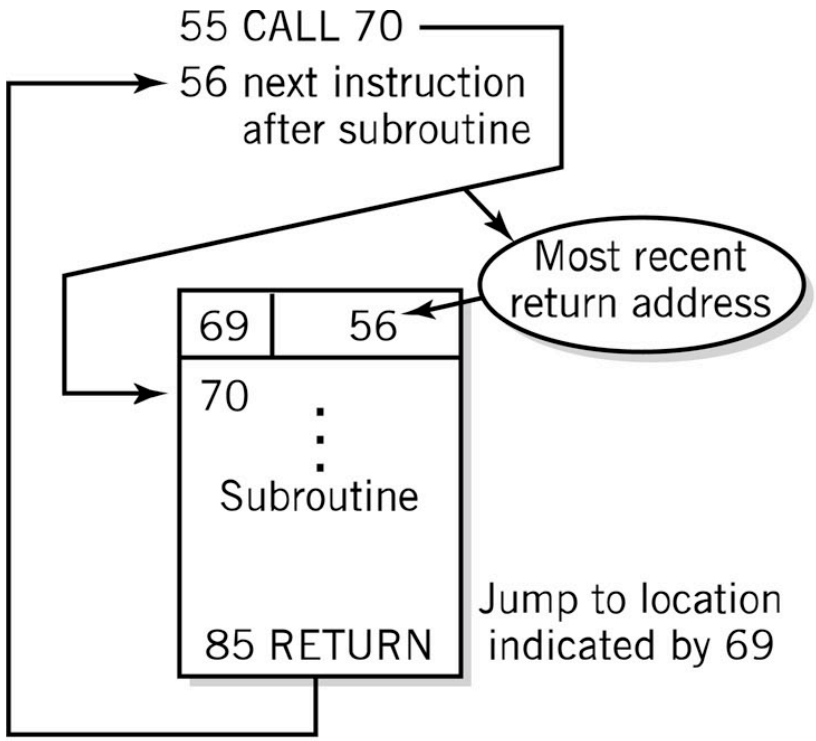
- LIFO méthode pour organiser l'information
- Items sont retirés dans l'ordre inversée de l'ordre d'arrivée



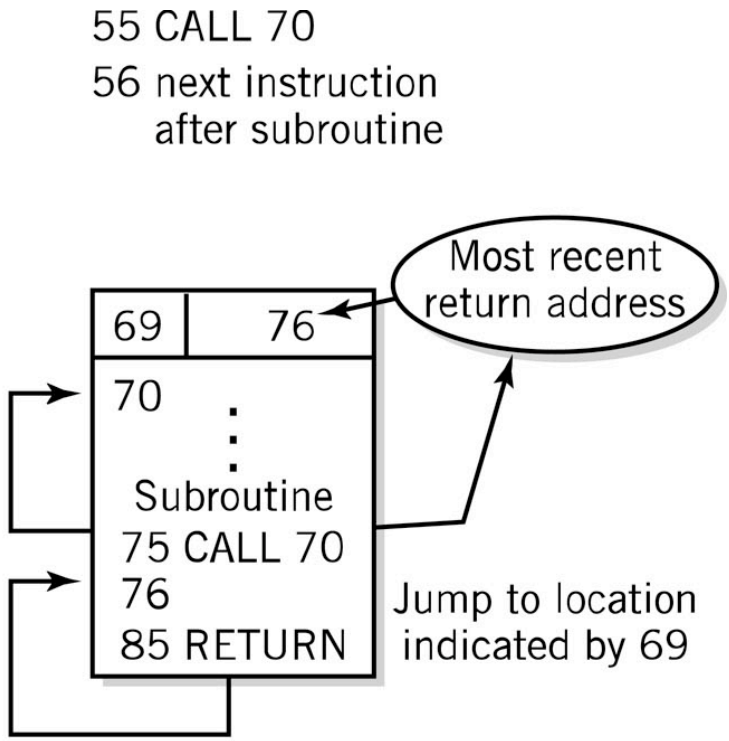
Empiler

Dépiler

Appels de sous-programmes

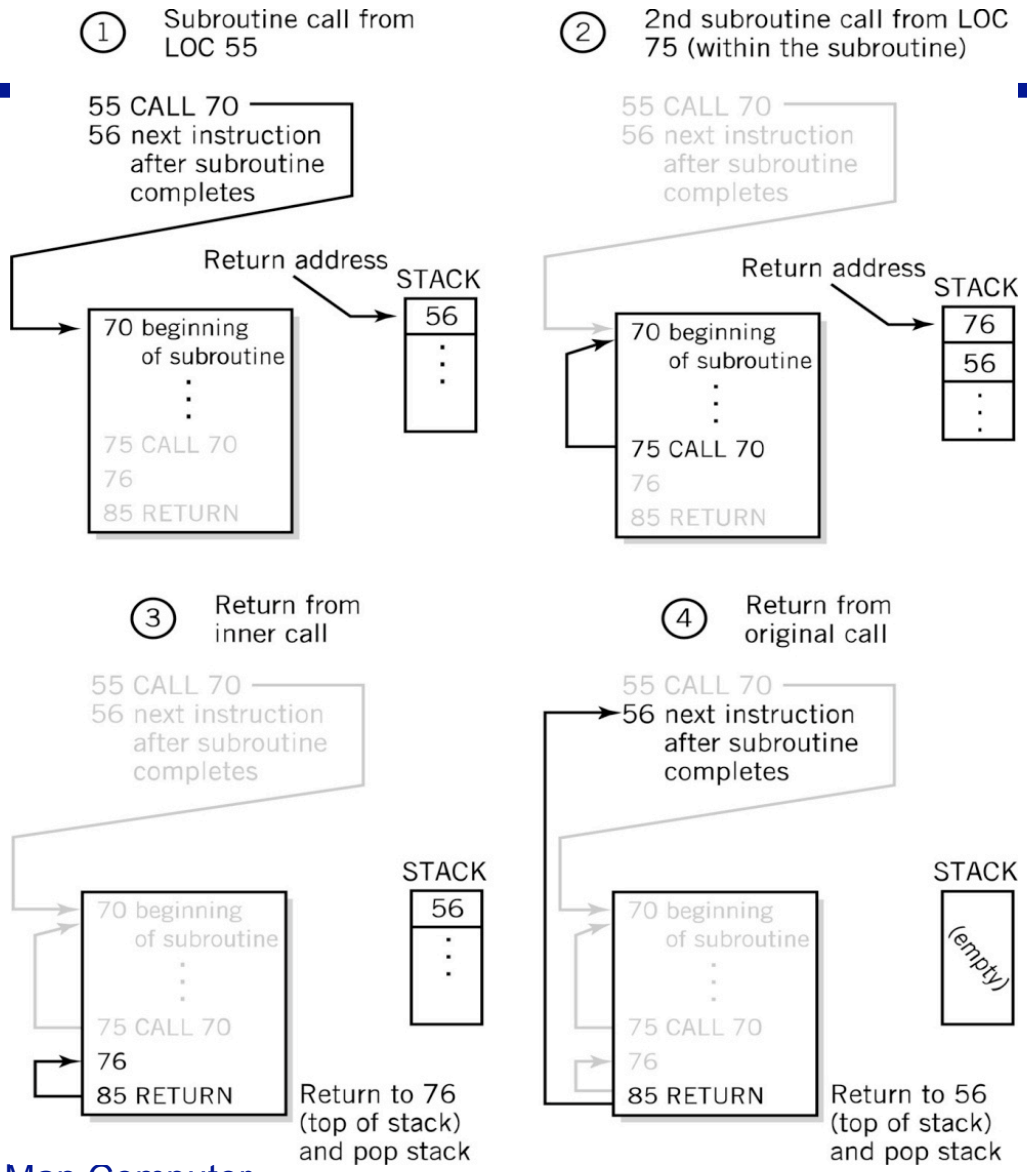


a. Subroutine called from loc.55

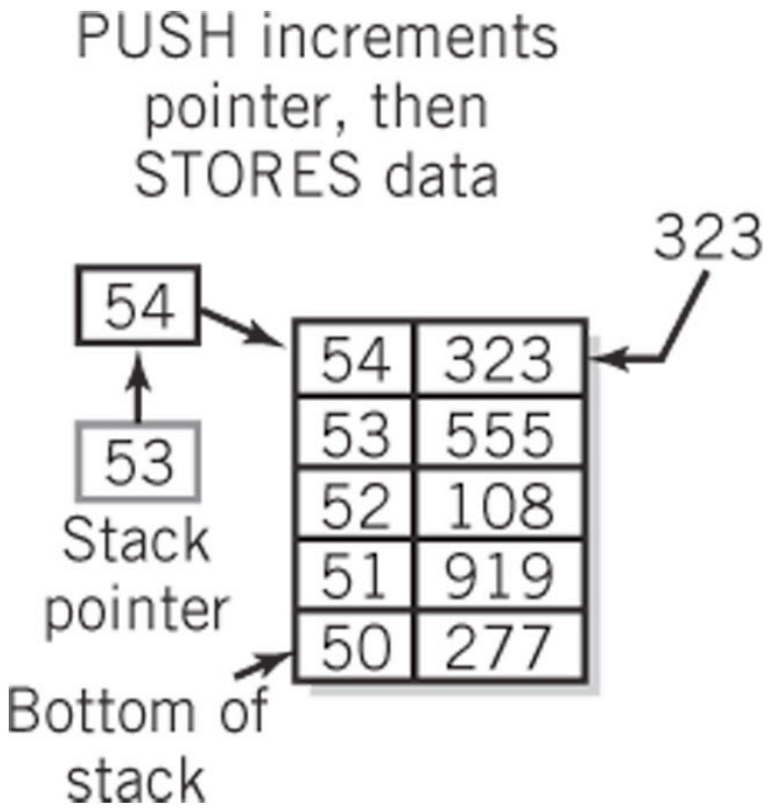


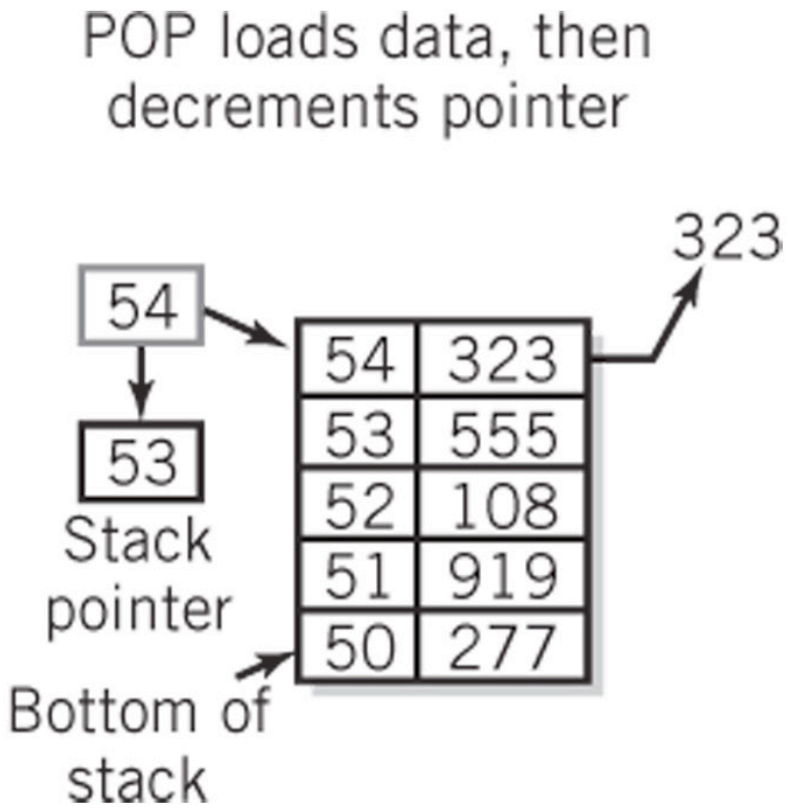
b. Subroutine re-called from 75, within the subroutine

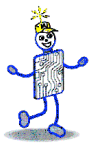
Appels de sous-programmes



Pile



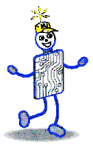




Éléments de l'instruction

- L'instruction machine est une chaîne binaire de p bits composée principalement de deux parties
  - Un code opération
    - Indique au processeur le type de traitement à réaliser
    - Un code opération de m bits permet de définir 2^m opérations différentes pour la machine
    - Le nombre d'opérations différentes autorisées pour une machine définit le jeu d'instructions de la machine

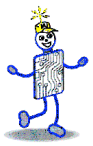
Code Opération	Champ opérandes



Le champ opérandes

- Composé de p-m bits
  - Indique la nature des données sur lesquelles l'opération désignée par le code opération doit être effectuée
    - Le façon de désigner un opérande dans une instruction peut prendre différentes formes
      - Mode d'adressage des opérandes

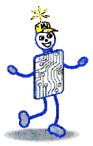
Code Opération	Champ opérandes



Types d'opérandes

- 1, 2 ou 3 opérandes (selon type d'instruction)
- 3 natures différentes (mode d'adressage)
  - L'opérande – une valeur immédiate, par ex. 3

Code Opération	Mode adressage immédiat	Information complémentaire = opérande = valeur immédiate = 3



Types d'opérandes

- L'opérande est un registre du processeur, par.ex. R1

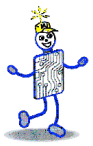
Code Opération	Mode adressage registre	Information complémentaire = = numéro de registre = 1

Opérande = contenu de R1

- L'opérande est un mot mémoire

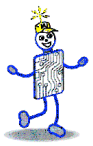
Code Opération	Mode adressage direct	Information complémentaire = = adresse mémoire = 128

Opérande = contenu de la case Mémoire 128 = 7



Le format du champ opérande

- Le mode d'adressage lié à l'opérande
- Une information complémentaire qui permet conjointement avec le mode d'adressage de trouver l'opérande



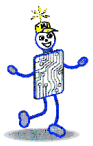
Éléments d'une instruction, résumé

- OPCODE (code opération): tâche
- Opérande(s) Source
- Opérande Résultat

Adresses

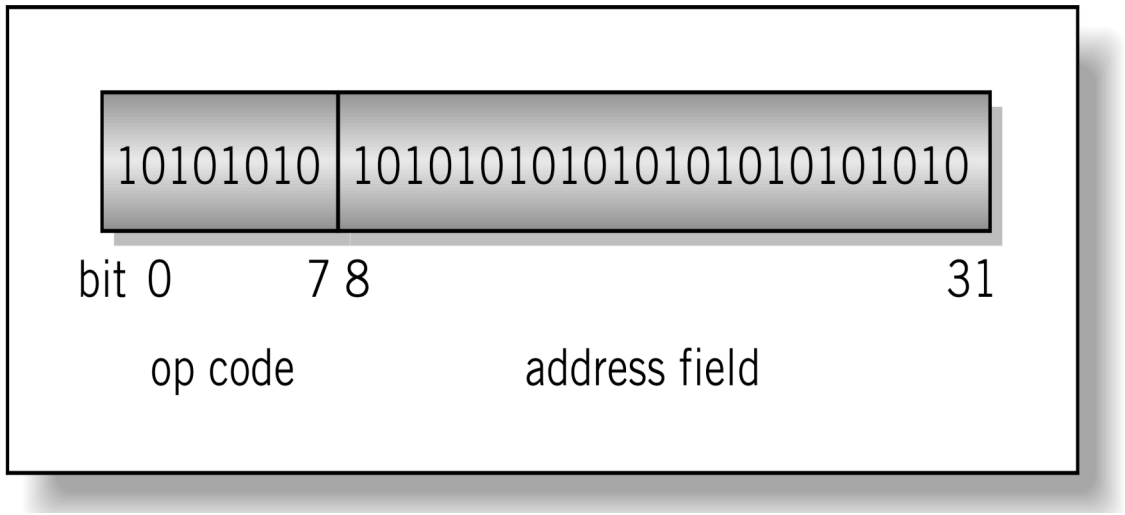
  - Emplacement de donnée (registre, mémoire)
    - Explicite: inclus dans l'instruction
    - Implicite: définit par default

OPCODE	OPÉRANDE Source	OPÉRANDE Résultat

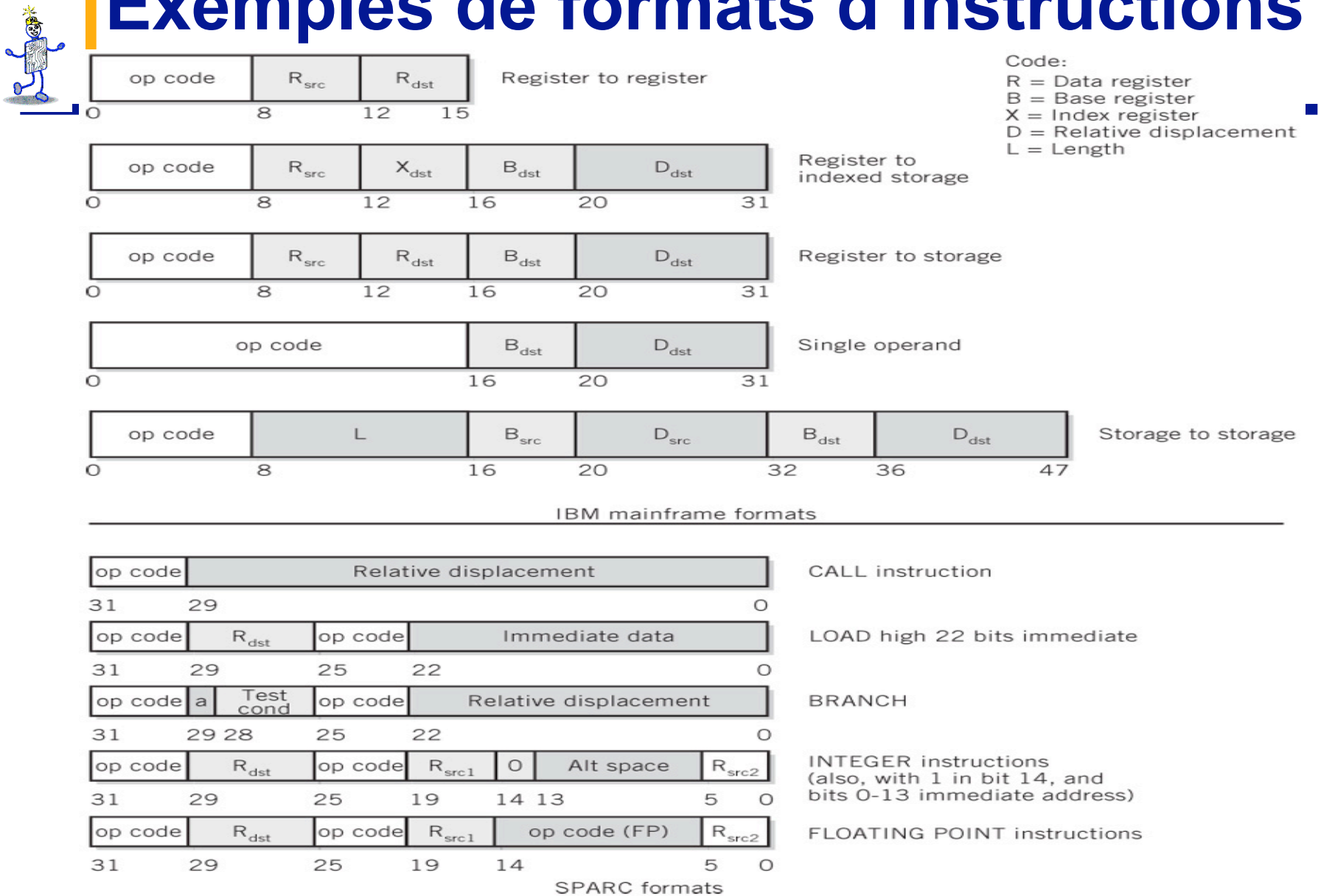


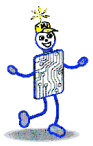
Format d'une instruction

- *Un gabarit spécifique à la machine* qui définit
  - Longueur de opcode
  - Nombre des opérandes
  - Longueur des opérandes



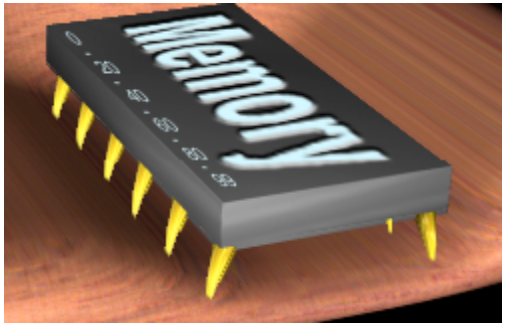
Exemples de formats d'instructions



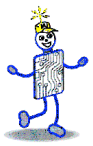


The Little Man Computer

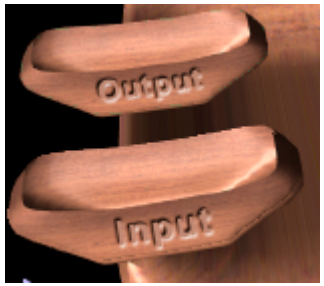
LMC, composants



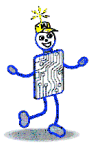
- Little Man – Le chef, il sait ce qu'il faut faire et il a autorité sur tous les autres
- Une armoire à tiroirs numérotés de 0 à 99. Chaque tiroir peut contenir un nombre de 0 à 999
- Le compteur de programme « Program Counter » : Il indique le numéro du tiroir de la prochaine instruction à réaliser



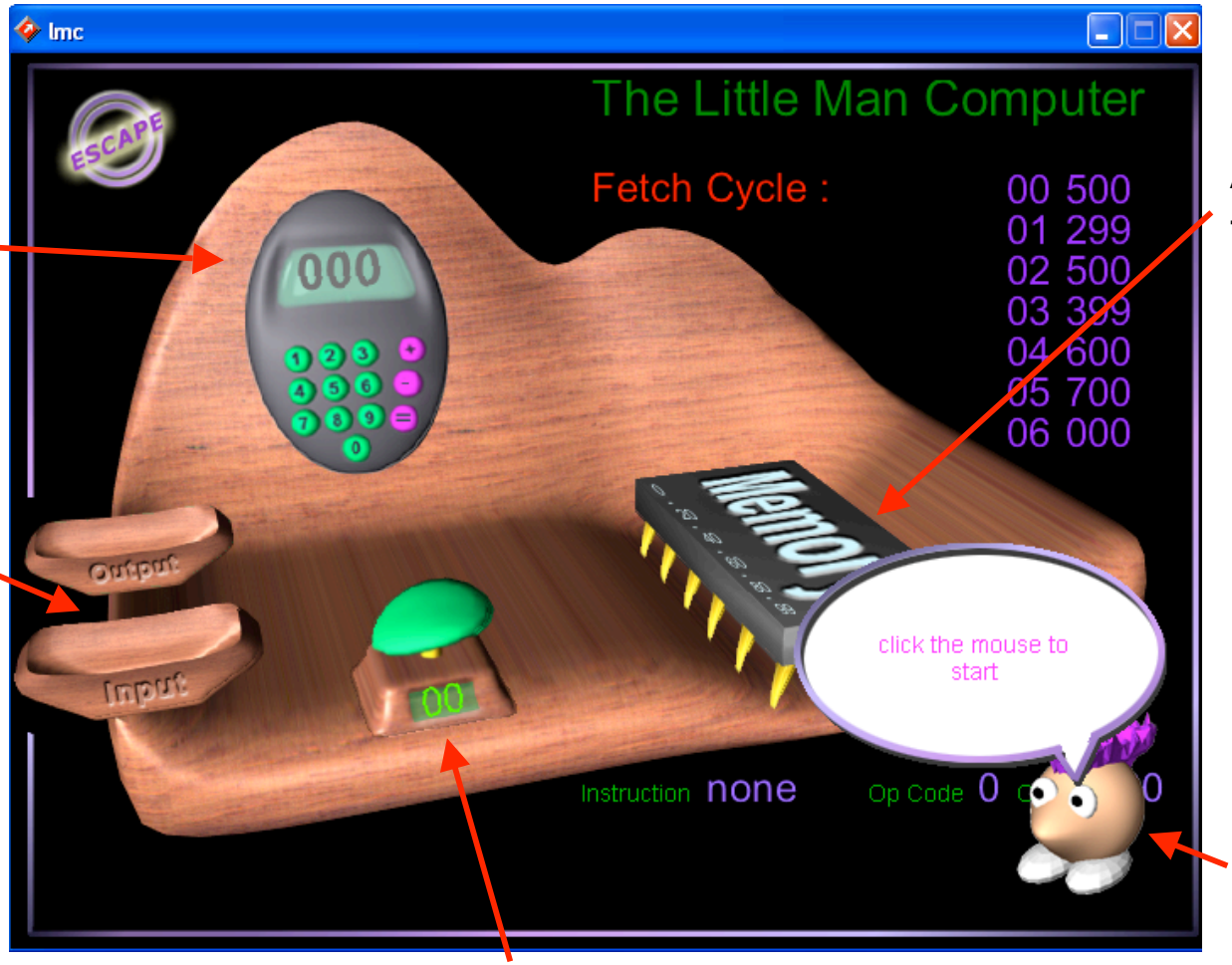
LMC



- L'unité de calcul. Elle est capable de faire des additions et des soustractions. Elle dispose d'une mémoire interne qui garde toujours le résultat de la dernière opération.
- Les corbeilles entrée/sortie . Le LMC est capable de recevoir et d'envoyer des nombre de 0 a 999



LMC



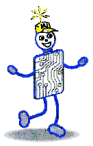
Unité de calcul

Armoire à tiroirs

Entrées/ Sorties

Little Man

Le compteur de programme



Armoire à tiroirs

- 100 tiroirs numérotés de 0 à 99 pouvant contenir des nombres de 0 à 999.
  - Un nombre de 0 à 999 est en fait 3 chiffres de 0 à 9
  - On peut donc voir la mémoire comme 100 tiroirs contenant chacun trois compartiments. Dans chaque compartiment, on peut mettre un chiffre de 0 à 9.

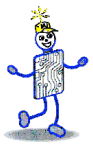
3	1	7

- En fonction du contexte, ces chiffres peuvent avoir différentes significations

Armoire à tiroirs: Adresse vs. Contenu

- Le langage machine du LMC est écrit en base 10
- Adresses sont consécutives
- Contenu peut être
  - Données ou
  - Instructions

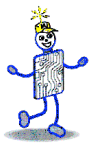
Adresse № du tiroir	Contenu



Contenu: Instructions LMC

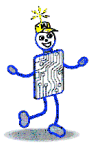
- Op code
  - Code opération, LMC – 1 digit
  - Mnémonique arbitraire
- Opérande
  - Objet pour la manipulation
    - LMC – 2 digits après op code
      - Adresse de données
      - Données

Adresse	Contenu	
	Op code	Opérande



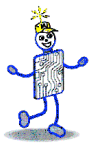
Magie!

- Charger le programme dans la mémoire
- Mettre les données dans un panier « in »



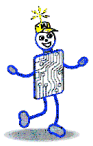
Langage d'assemblage

- Spécifique au CPU
- Langage d'assemblage – une variante symbolique du langage machine
  - Correspondance 1 à 1
- *Mnémoniques* (courte séquence de caractères) représentent les instructions
- Utilisé quand le programmeur besoin un contrôle précis sur le matériel (pilotes)



Jeu d'Instructions

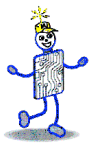
Arithmétiques	1xx	ADD
	2xx	SUB
Transfert de données	3xx	STORE
	5xx	LOAD
Entrée/Sortie	901	INPUT
	902	Output
Contrôle de la machine (coffee break)	000	HALT
		COB



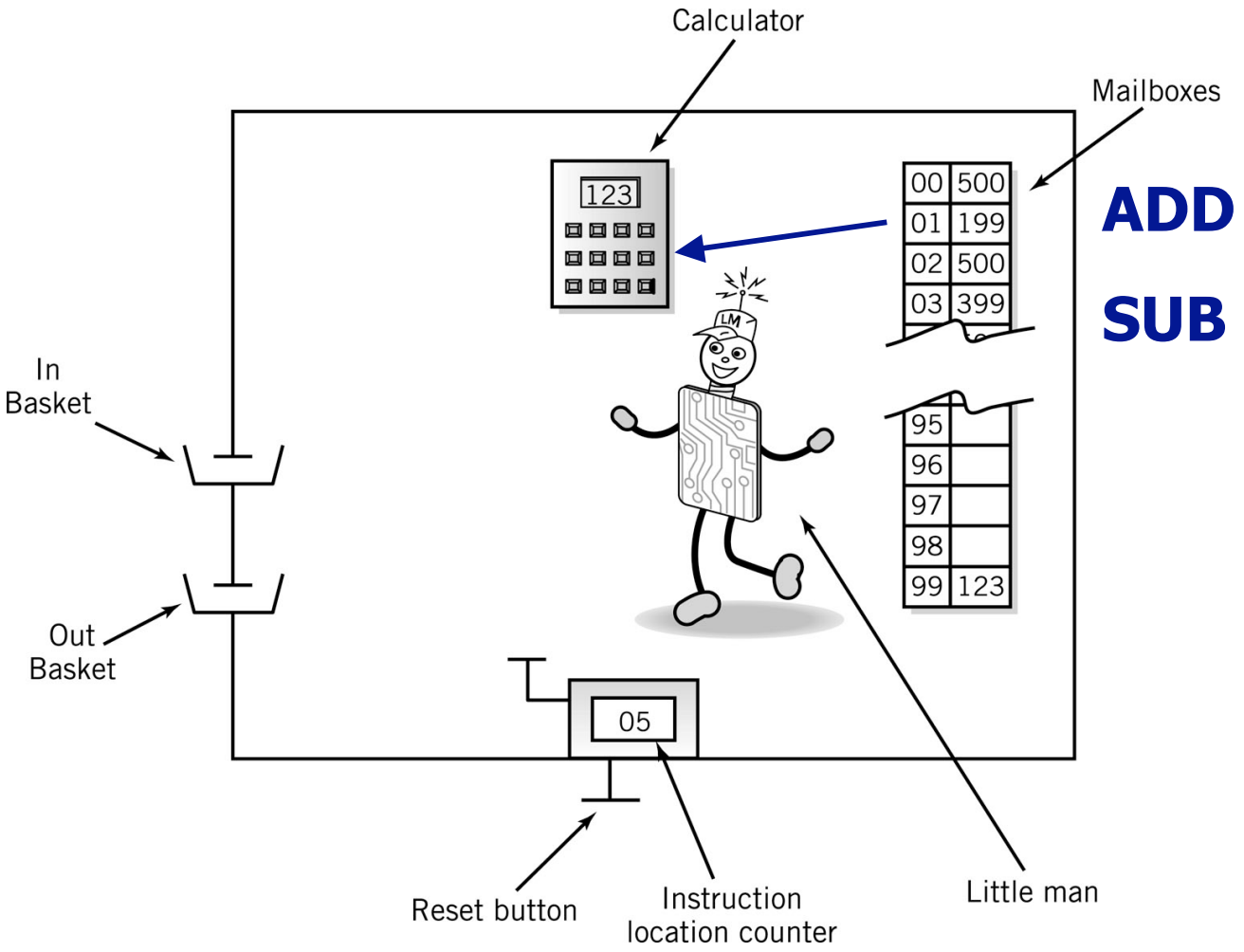
Les Instructions arithmétiques

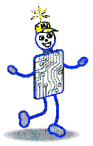
- Lire le contenu du tiroir
- Faire l'opération dans l'unité de calcul

	Contenu	
	Op Code	Opérande (adresse)
ADD	1	xx
SUB	2	xx



LMC, Instructions arithmétiques

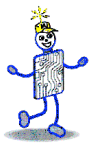




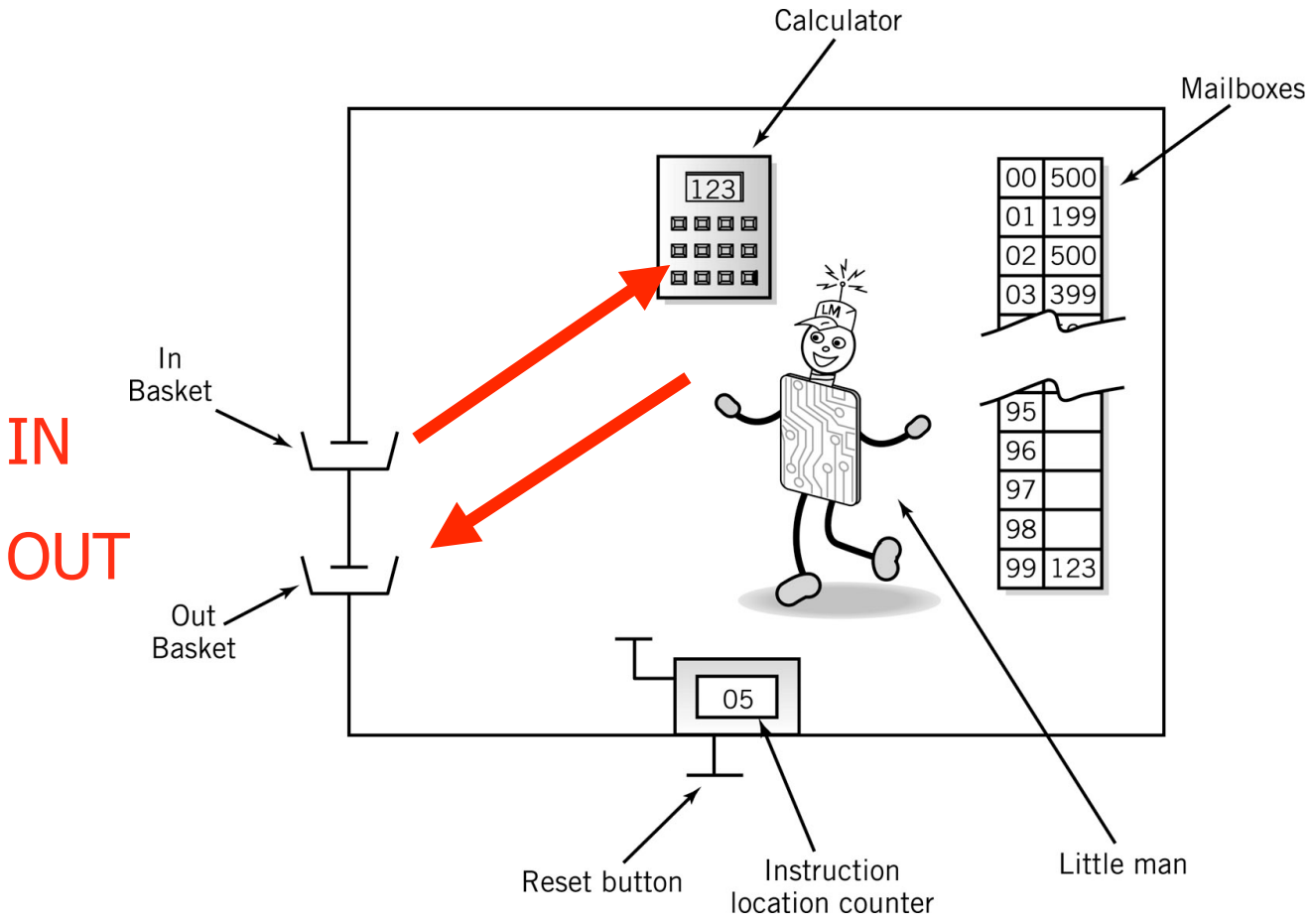
Entrée/Sortie

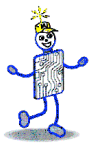
- Transfert de données depuis l'unité de calcul vers les entrées/sorties et vice versa

	Contenu	
	Op Code	Opérande (adresse)
IN (input)	9	01
OUT (output)	9	02



LMC Entrée/Sortie

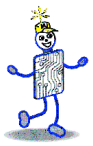




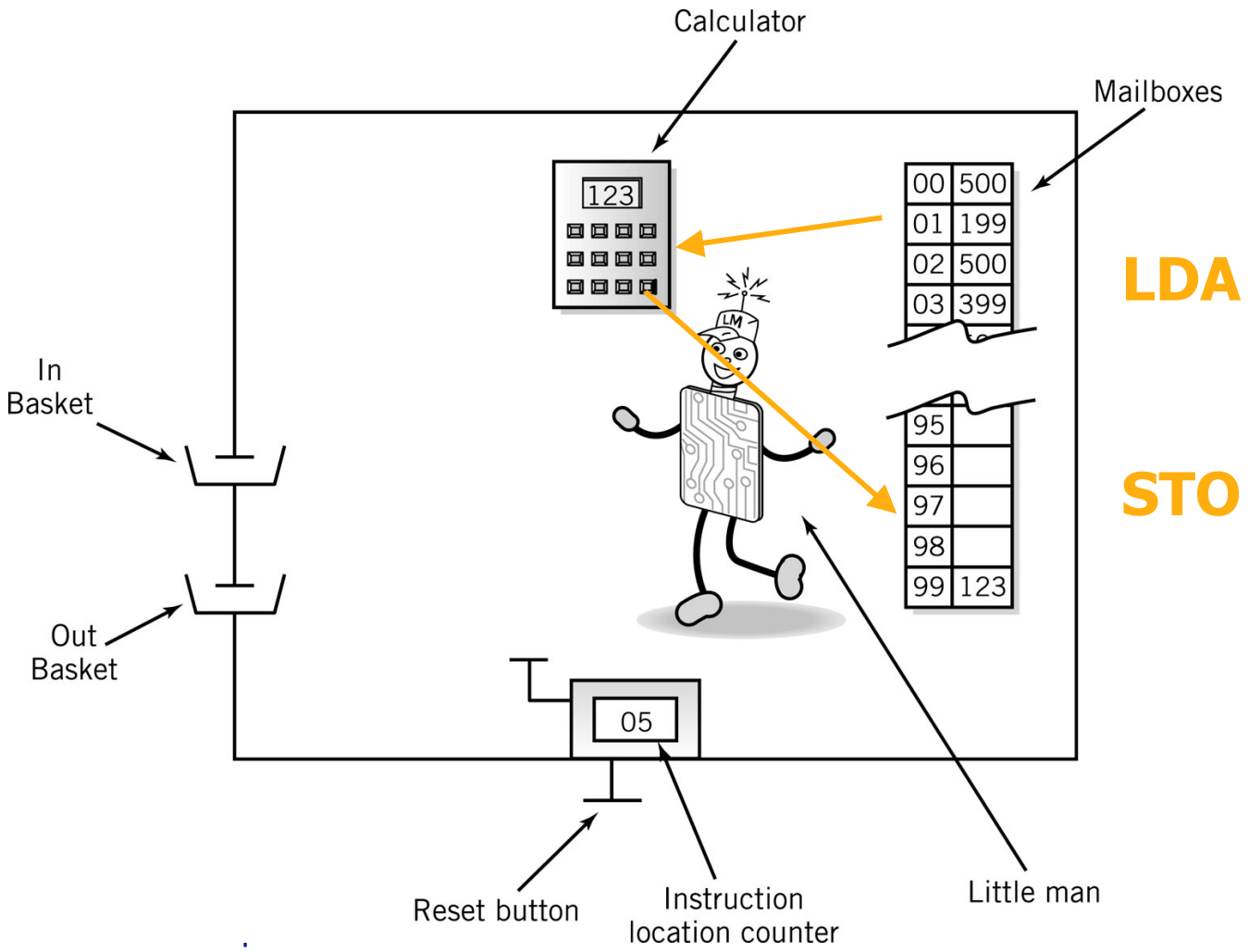
Transfert de données

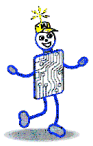
- Entre tiroirs et l'unité de calcul

	Contenu	
	Op Code	Opérande (adresse)
STO (store)	3	xx
LDA (load)	5	xx



LMC, Transfert de données

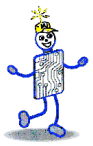




Données

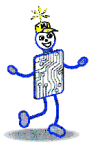
- Identiques aux instructions
- Ne doivent pas être placées dans la séquence d'instructions
- Identifiées par mnémonique *DAT*
- Pseudo-instructions
  - Ordres destinés au traducteur assembleur

```
DAT     003
```



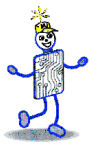
Langage d'assemblage

- Programmation en langage d'assemblage nécessite une étape de traduction
  - Les instruction en langage machine sont compréhensibles et exécutables par la machine
- Phase de traduction
  - Un outil appelé l'assembleur



Langage d'assemblage

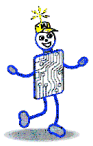
- Format d'une instruction du langage d'assemblage
  - Une instruction du langage d'assemblage est composée de champs, séparés par un ou plusieurs espaces
    - Champ étiquette
    - Champ code opération
    - Champ opérandes
      - Plusieurs opérandes séparés par des virgules
    - Champ commentaires



Langage d'assemblage

- Format d'une instruction du langage d'assemblage

Étiquette	Code opération	Opérandes	Commentaires



Étiquette

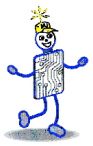
- Une chaîne de caractères permettant de nommer une instruction ou une variable

Étiquette	Code opération	Opérandes	Commentaires

- Correspond à une adresse dans le programme
  - Instruction

    loop LDA var
  - Variable

    var DAT 000

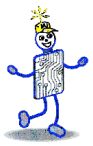


Langage d'assemblage LMC

- Code opérations

Étiquette	Code opération	Opérandes	Commentaires

  - Une chaîne de caractères mnémonique du code opération
    - LDA
    - STO
    - IN
    - OUT
    - ADD



Langage d'assemblage LMC

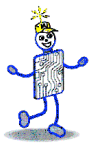
- Les opérandes

Étiquette	Code opération	Opérandes	Commentaires

  - Adresse de l'opérande (mode d'adressage direct)
    - Étiquette

```
ADD     one
…
one     DAT     001
```

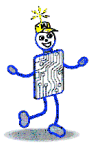
```
ADD     99
…
99      DAT     001
```



L'Arithmétique signée en base 10 (LMC)

- Nombres signés
  - Convention complément à 10
  - Avec 3 compartiments dans chaque tiroir et un langage en base 10 => 000 - 999
    - tous les nombres >=500 sont considérés comme étant négatifs

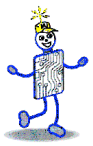
500	**-**	**999**	**0**	**+**	**499**



L'Arithmétique signée en base 10 (LMC)

- Nombres signés
  - Convention complément à 10
    - Nombres positif [0,499]
    - Nombres négatif [-1, -500]
      - Complément à 9 de la valeur absolue
      - Ajouter 1

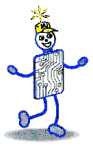
500	**-**	**999**	**0**	**+**	**499**



L'Arithmétique signée en base 10 (LMC)

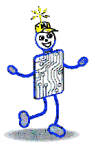
- Nombres signés
  - Nombres négatif [-1, -500]
  - Exemple: -347
    - Complément à 9 de la valeur absolue
      - C–à-9 sur 3 chiffres de 347 = 652
    - Ajouter 1
      - 653

500 - 999	0 + 499
-347	



Additionner deux nombres

- Réalisez un programme qui additionne deux nombres ensemble avec le jeux d'instructions vu précédemment
  - Lire la première entrée (*Input)*
  - Lire la deuxième entrée (*Input*)
  - Additionner les deux (*Add*)
  - Envoyer le résultat (*Output*)



Additionner deux nombres

- Problèmes :
- *Input* écrit la valeur dans l'unité de calcul et il n'y a qu'une place.
- *Add* calcule la somme du contenu d'un tiroir avec le contenu courant de l'unité de calcul

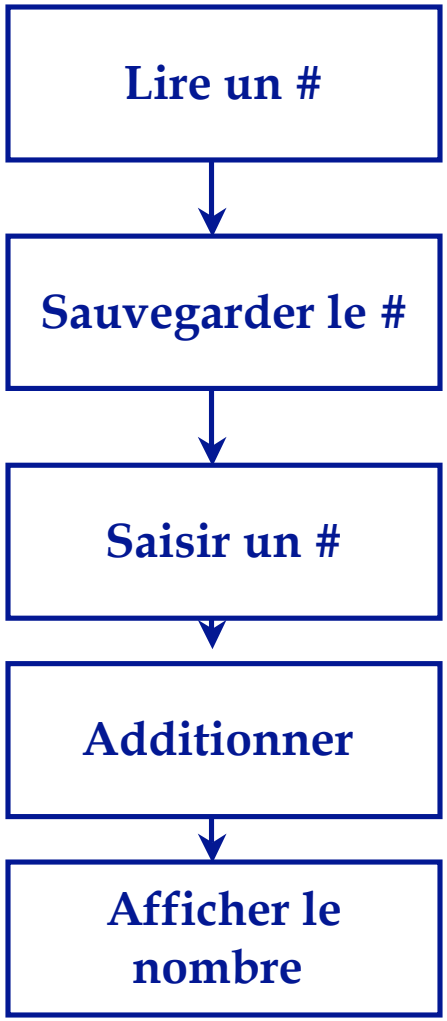
Additionner deux nombres

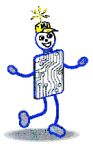
- Il faut donc mémoriser temporairement la première donnée lue
- Le programme devient:
  - 1) Lire la première entrée (*Input)*
  - 2) Écrire cette donnée en mémoire
  - 3) Lire la deuxième entrée (*Input*)
  - 4) Additionner le contenu de l'unité de calcul avec le contenu du tiroir (*Add*)
  - 5) Envoyer le résultat (*Output*)

Additionner deux Nombres

- Quel tiroir utiliser pour sauvegarder la donnée ?
  - Données sont stockées dans les tiroirs

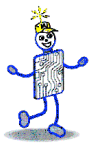
    avec les adresses >90





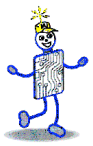
Programme d'Addition de 2 Nombres: en mnémoniques

N° tiroir	Mnémonique	Description
00	IN	;input 1st Number
01	STO 99	;store data
02	IN	;input 2nd Number
03	ADD 99	;add 1st # to 2nd #
04	OUT	;output result
05	HLT	;stop
99	DAT 00	;data



Programme d'Addition de 2 Nombres

N° tiroir	Contenu du tiroir	Description
00	901	;input 1st Number
01	399	;store data
02	901	;input 2nd Number
03	199	;add 1st # to 2nd #
04	902	;output result
05	000	;stop
99	000	;data



Contrôle

- Branchement (les instructions de rupture de séquence d'exécution)
  - Change l'adresse de l'instruction à exécuter
- Arrêt du processeur (Halt)

BR (Jump)

BRZ (Branch on 0)

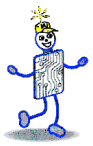
BRP (Branch on +)

COB (stop)

Contenu	
Op Code	Opérande (adresse)
6	xx
7	xx
8	xx
0	(ignorée)

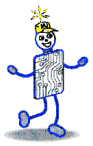
Contrôle

- Branchement
  - Instruction de sauts inconditionnels
    - Effectue toujours le débranchement de l'exécution à l'adresse spécifiée
    - BR XX (XX – adresse de branchement)
  - Instructions de sauts conditionnels
    - Effectuent le débranchement de l'exécution si et seulement si une condition correspondante est vérifiée
    - BRZ XX (si le contenu de la calculatrice = 0, on fait le saut à l'adresse XX)
    - BRP XX (si le contenu de la calculatrice > ou = 0, on fait le saut à l'adresse XX)



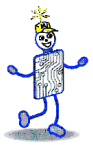
Jeu d'instructions LMC

Arithmétiques	1xx	ADD
	2xx	SUB
Transfert de données	3xx	STO
	5xx	LDA
Branchement	6xx	BR
	7xx	BRZ
	8xx	BRP
Entrée/Sortie	901	IN
	902	OUT
Contrôle de la machine	000	HLT



Trouver une différence positive de 2 nombres. Langage d'assemblage LMC

	IN	
	STO D1	
	IN	
	STO D2	
	SUB D1	
	BRP AF	;test
	LDA D1	;if negative, reverse order
	SUB D2	
AF	OUT	;print result and
	HLT	;stop
D1	DAT 00	;used for data
D2	DAT 00	;used for data



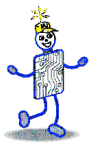
Trouver une différence positive de 2 nombres. Langage machine LMC

Mémoire

00	901
01	310
02	901
03	311
04	210
05	808
06	510
07	211
08	902
09	000
10	000
11	000

Exécution d'un programme

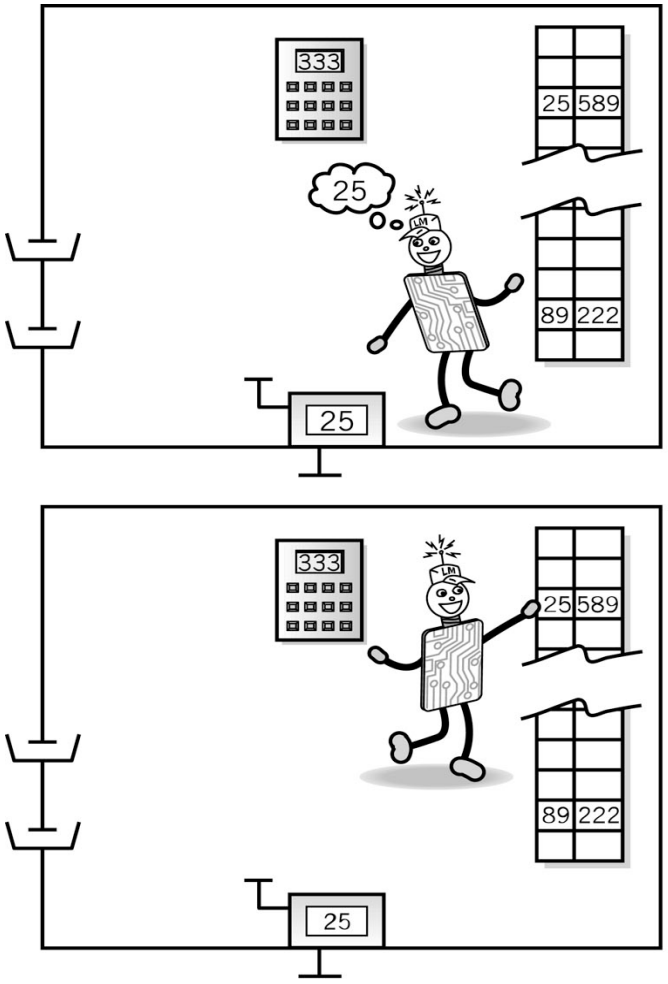
- Pour exécuter un programme selon le modèle LMC, on doit suivre les étapes
  - Charger les instructions du programme dans les tiroirs en partant du tiroir no 00
  - Placer la (les) donnée(s) qui sera utilisée(s) par le programme dans le panier « IN » (dans l'ordre où le programme les utilisera)
  - Presser le bouton RESET pour initialiser le compteur d'instructions à 00 et avertir le LMC qu'un programme doit être exécuté



Cycle d'instruction

- Différentes phases de réalisation des instructions
  - *Fetch* (recherche de l'instruction) Little Man trouve l'instruction à exécuter
  - *Execute*: Little Man exécute l'instruction.

Étape « Fetch »

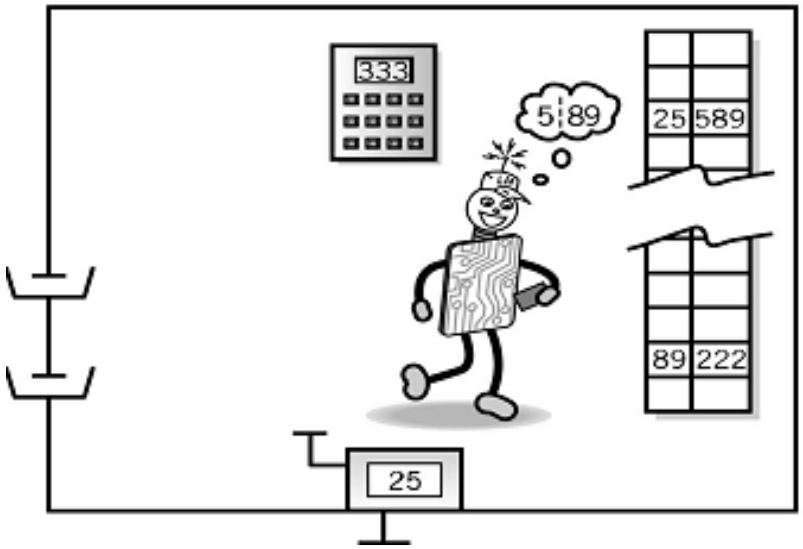


1. Lire le compteur de programme pour savoir dans quel tiroir se trouvent les chiffrent qui codent

l'instruction à exécuter.

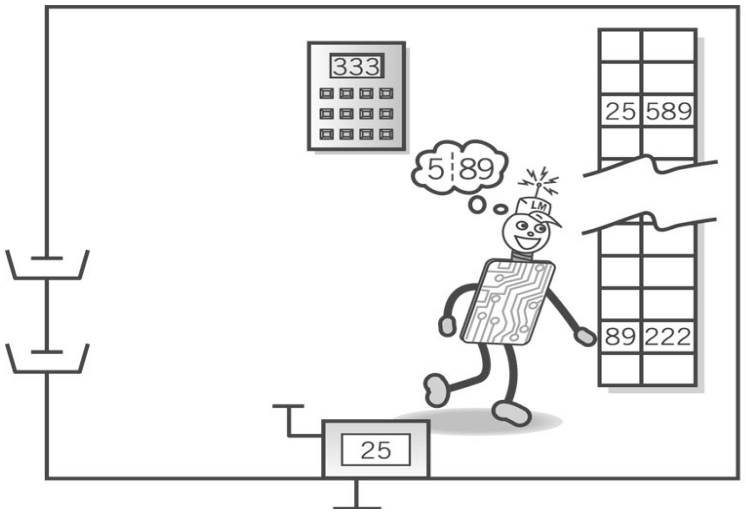
2. LM va au tiroir

Fetch, cont.



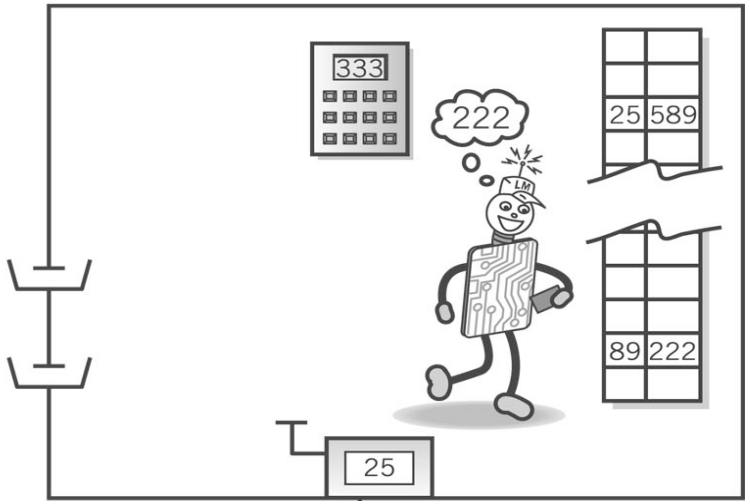
3. Lire les chiffres qui se trouvent dans le tiroir concerné. (FETCH)

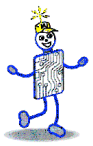
Étape “Exécuter”



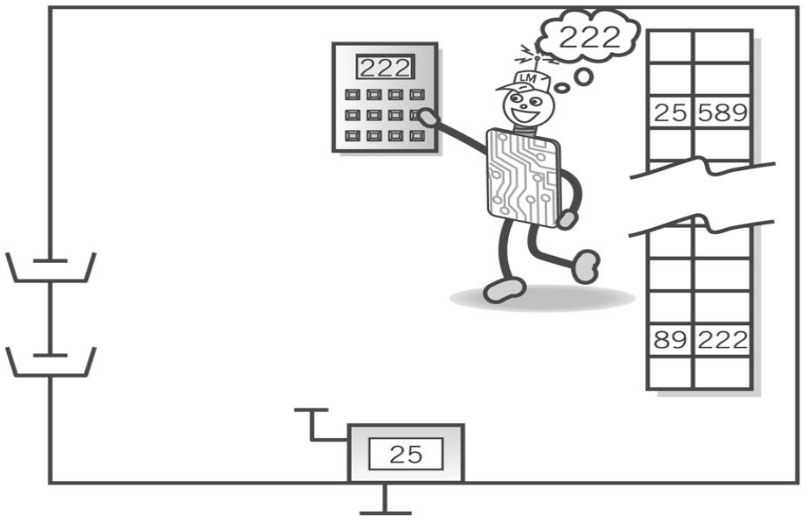
1. Little Man Regarde de quelle instruction s’agit le code (DECODE) et cherche l’opérande

2. Il lit l’opérande.



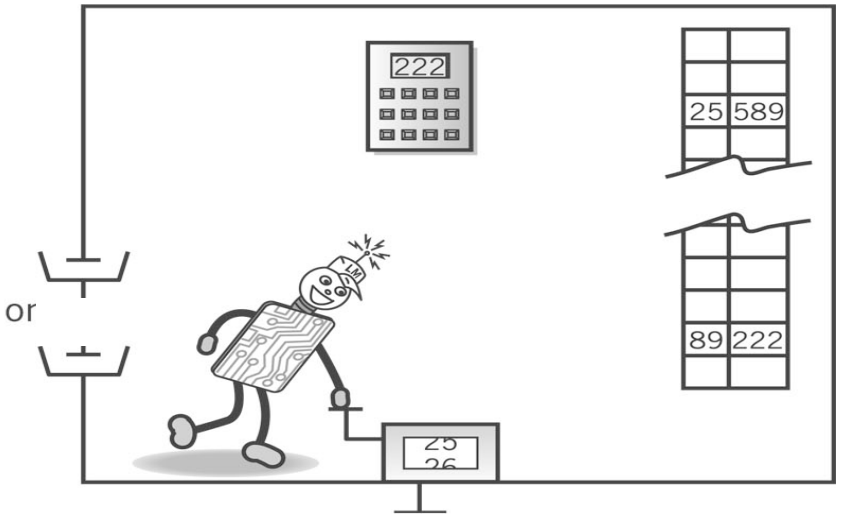


Exécuter



3. Exécute l'opération et place le résultat dans l'unité de calcul

4. Incrémenter (faire +1) le compteur de programme.



Architecture Von Neumann (1945)

- L'architecture des ordinateurs reste virtuellement inchangée depuis 1951, alors que la technologie des composants évolue si vite
- Concepts clés de l'architecture de Von Neumann
  - Concept de programme stocké en mémoire
    - Mémoire stockant les données et le programme

Architecture Von Neumann (1945)

- La mémoire est adressée linéairement
  - Adresse numérique séquentielle unique pour chaque espace mémoire
- Chaque espace mémoire possède une adresse et un contenu tous deux étant différents
- Les instructions s'exécutent linéairement à moins d'une instruction spécifique de branchement