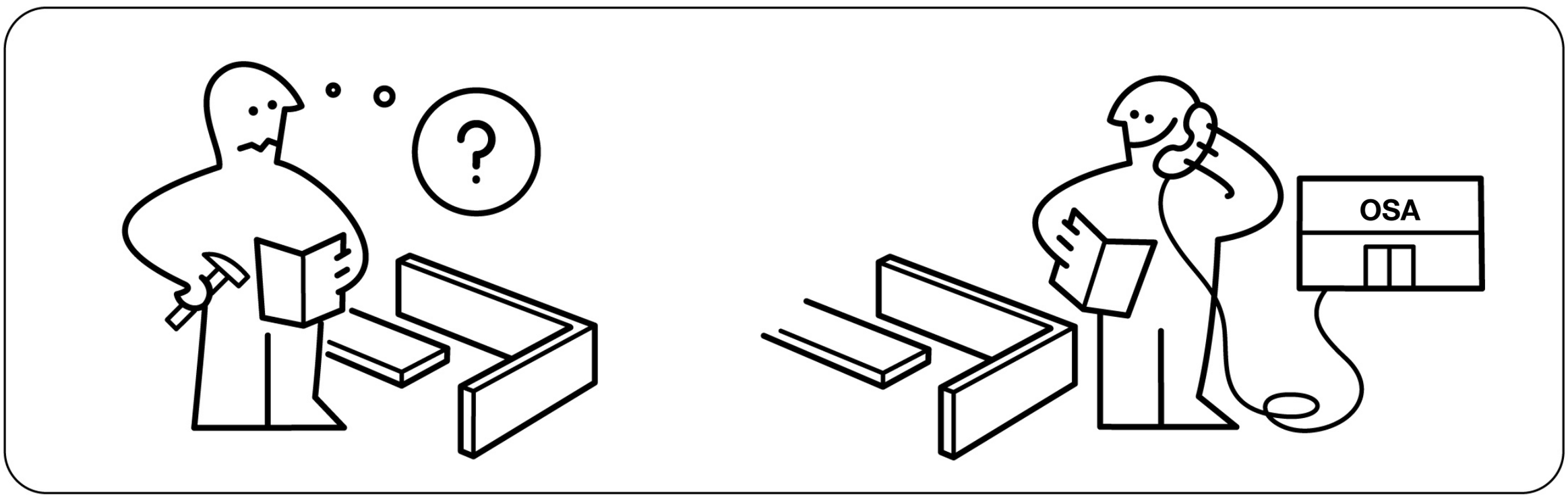
Instructions et jeu d'instructions

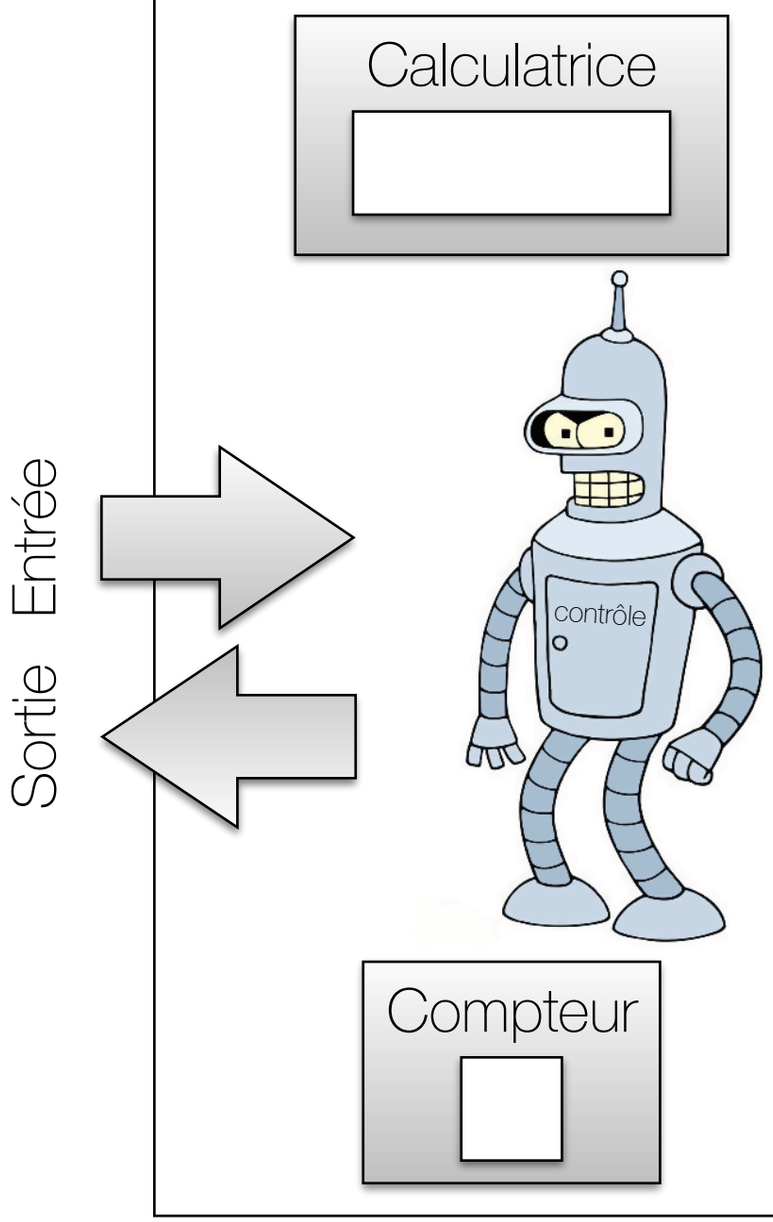


Ordinateurs: Structure et Applications, Hiver 2016

Jean-François Lalonde

Rappel: ordinateur simplifié

Ordinateur



Mémoire

Adresse	Contenu
00	901
01	309
02	901
03	109
04	902
05	000
06	
07	
08	
09	

Liste des instructions disponibles

1xx	ADD additionne le nombre à l'adresse xx à la calculatrice
2xx	SUB soustrait le nombre de l'adresse xx à la calculatrice
3xx	STORE stocke le contenu de la calculatrice à l'adresse xx
5xx	LOAD charge le contenu de l'adresse xx dans la calculatrice
901	INPUT lis l'entrée et l'écrit dans la calculatrice
902	OUTPUT écrit le contenu de la calculatrice dans la sortie
000	BREAK arrête l'exécution

Instructions et jeu d'instructions

- Une instruction est une « action » pouvant être exécutée par le microprocesseur.
- Un « jeu d'instructions » représente toutes les instructions pouvant être exécutées par un microprocesseur.
- Il existe plusieurs types d'instructions: mouvements de données, arithmétique et logique, contrôle de programme, etc.

Instructions (ordinateur simplifié)

Jeu d'instructions

1xx	ADD additionne le nombre à l'adresse xx à la calculatrice	Opérations arithmétiques
2xx	SUB soustrait le nombre de l'adresse xx à la calculatrice	
3xx	STORE stocke le contenu de la calculatrice à l'adresse xx	Mouvement de données
5xx	LOAD charge le contenu de l'adresse xx dans la calculatrice	
901	INPUT lis l'entrée et l'écrit dans la calculatrice	
902	OUTPUT écrit le contenu de la calculatrice dans la sortie	
000	BREAK arrête l'exécution	Flot du programme

Instructions (TP1)

Jeu d'instructions

Mnémonique	
MOV Rd Rs	Déplacement de données
MOV Rd Const	
ADD Rd Rs	Opérations arithmétiques
ADD Rd Const	
SUB Rd Rs	
SUB Rd Const	
LDR Rd [Rs]	Déplacement de données
STR Rd [Rs]	
JZE Rc Const	Flot du programme
JZE Rc [Rs]	

Déplacement de données

- D’un registre à l’autre

Instructions (TP1)	Signification
`MOV R1 R2`	`R1 ← R2`
`MOV R0 #0x71`	`R0 ← 0x71`

- est-ce qu’on accède à la mémoire?

Déplacement de données

- De la mémoire vers un registre

Instructions (TP1)	Signification
`LDR R1 [R2]`	`R1 ← Memoire[R2]`

  - lecture ou écriture?

- D'un registre vers la mémoire

Instructions (TP1)	Signification
`STR R1 [R2]`	`Memoire[R2] ← R1`

  - lecture ou écriture?

`Memoire[XX] =`
contenu de la mémoire à l'adresse 'xx'

Opérations arithmétiques

- Additions ou soustractions de nombres entiers

Instructions (TP1)	Signification
`ADD R1 R2`	`R1 ← R1 + R2`
`ADD R1 #0x2`	`R1 ← R1 + 0x2`

- Multiplication ou divisions

Instructions (ARM)	Signification
`MUL R0 R1 R2`	`R0 ← R1 x R2`

Contrôle de programmes

- « Sauter » d'une adresse mémoire à un autre

Instructions (TP1)	Signification
`JZE R0 [R1]`	`Si R0 == 0, alors PC ← R1`
`JZE R0 #0x2`	`Si R0 == 0, alors PC ← 0x2`

- Appeler une fonction

Instructions (TP1)	Signification
`BL nomDeFonction`	`PC ← Adresse de nomDeFonction`

  - plus de détails dans 2 semaines

Rotations et décalages binaires

- Décalage (« shift »)
  - “tasser” tous les bits vers la droite ou la gauche
  - quelle est l’opération arithmétique correspondante?
    - que faire avec les nombres entiers en complément 2?
- Rotation
  - comme un décalage, sauf qu’on replace le bit à droite (ou à gauche) au lieu d’insérer un 0 (ou un 1).

Instructions SIMD

- SIMD: “Single Instruction, Multiple Data”
- Traiter plusieurs données en même temps, particulièrement utile pour des applications multimédias
  - ex: image = vecteur de pixels. On veut souvent appliquer la même opération (single instruction) à tous les pixels (multiple data)

Question

- Comment représente-t-on une instruction dans un ordinateur?
- En binaire, pardi!

Structure d'une instruction

- Instruction:
  - code d'opération ("opcode") en binaire
  - des paramètres: format et taille dépendent de l'opcode
- La taille et le format d'une instruction peuvent varier
- Par exemple (TP1):
  - instruction sur 16 bits
  - 4 premiers bits: opcode
    - combien d'opcodes peut-on définir au total?
  - 12 derniers bits: paramètres

Opcode	Argument 1	Argument 2
4 bits	4 bits	8 bits

Jeu d'instructions

- La table ci-dessous est un exemple de jeu d'instructions.

- Chaque instruction possède un mnémonique en assembleur.

Toutes les instructions du microprocesseur sont sur 16 bits et se décomposent comme suit :
Bits 15 à 12 : Opcode de l'instruction
Bits 11 à 8 : Registre utilisé comme premier paramètre.
Bits 7 à 0 : Registre ou constante utilisés comme deuxième paramètre

Le microprocesseur possède quatre registres généraux nommés R0, R1, R2 et R3. En plus de ces quatre registres, un registre de pointeur d'instruction PC est disponible. Cependant, ce registre ne peut être utilisé qu'avec l'instruction MOV. Le nombre identifiant le registre PC est 0xF (15).

Le jeu d'instruction supporte les instructions suivantes où Rd est le registre destination, Rs le registre source et Rc le registre de condition :

Mnémonique	**Opcode**	**Description**
MOV Rd Rs	0000	Écriture de la valeur du registre Rs dans le registre Rd
MOV Rd Const	0100	Écriture d'une constante dans le registre Rd
ADD Rd Rs	0001	Addition des valeurs des registres Rd et Rs et insertion du résultat dans le registre Rd
ADD Rd Const	0101	Addition de la valeur du registre Rd avec une constante et insertion du résultat dans Rd
SUB Rd Rs	0010	Soustraction de la valeur Rs à l'intérieur de registre Rd.
SUB Rd Const	0110	Soustraction d'une constante à l'intérieur du registre Rd
LDR Rd [Rs]	1000	Chargement d'une valeur se trouvant à l'adresse Rs de l'ordinateur dans un registre.
STR Rd [Rs]	1001	Écriture de la valeur d'un registre à l'adresse Rs de l'ordinateur.
JZE Rc Const	1111	Saut à l'instruction située à l'adresse identifiée par la constante, mais seulement si Rc = 0 (sinon, cette instruction n'a aucun effet).
JZE Rc [Rs]	1011	Saut à l'instruction située à l'adresse Rs seulement si Rc = 0 (sinon, cette instruction n'a aucun effet).

RISC & CISC

- Il existe plusieurs approches pour la conception d'un microprocesseur et de son jeu d'instructions. Ces approches influencent chaque aspect du design de l'architecture d'un microprocesseur. Les principales approches utilisées à ce jour sont CISC et RISC.

- CISC (Complex Instruction Set Computer)

    - jeu d'instructions complexe dont la longueur (des instructions) varie. Comme les instructions peuvent être longues et complexes, peu de registres sont requis.

    - Exemple: x86: (8086, Pentium)

- RISC (Reduced Instruction Set Computer)

    - jeu d'instructions simple dont la longueur est fixe (ex: 4 octets). Plusieurs registres sont requis pour exécuter des tâches complexes.

    - Exemple: PowerPC, ARM

RISC vs. CISC

Avantages RISC (**R**educed **I**nstruction **S**et **C**omputer)	**Avantages CISC** (**C**omplex **I**nstruction **S**et **C**omputer)
10 instructions sont utilisées 70% du temps	Plus de flexibilité au programmeur (par exemple, transferts mémoire-mémoire)
Avoir plusieurs registres permet d'éviter les accès mémoires (qui sont plus lents)	Programmes plus courts, plus petits
Opérations "fetch-decode-execute" sont simplifiées, car toutes les instructions ont la même taille	
Opérations simplifiées = architecture simplifiée = consommation réduite	

- Les microprocesseurs modernes sont des microprocesseurs RISCs ou hybrides (un microprocesseur supportant des instructions ayant deux longueurs seulement par exemple).
- Les microprocesseurs CISCs disponibles découpent habituellement les instructions complexes en instructions simples (comme du RISC) avant de les exécuter.