Principes de fonctionnement des machines binaires

2019/2020

Pierluigi Crescenzi
Université de Paris, IRIF

Université de Paris

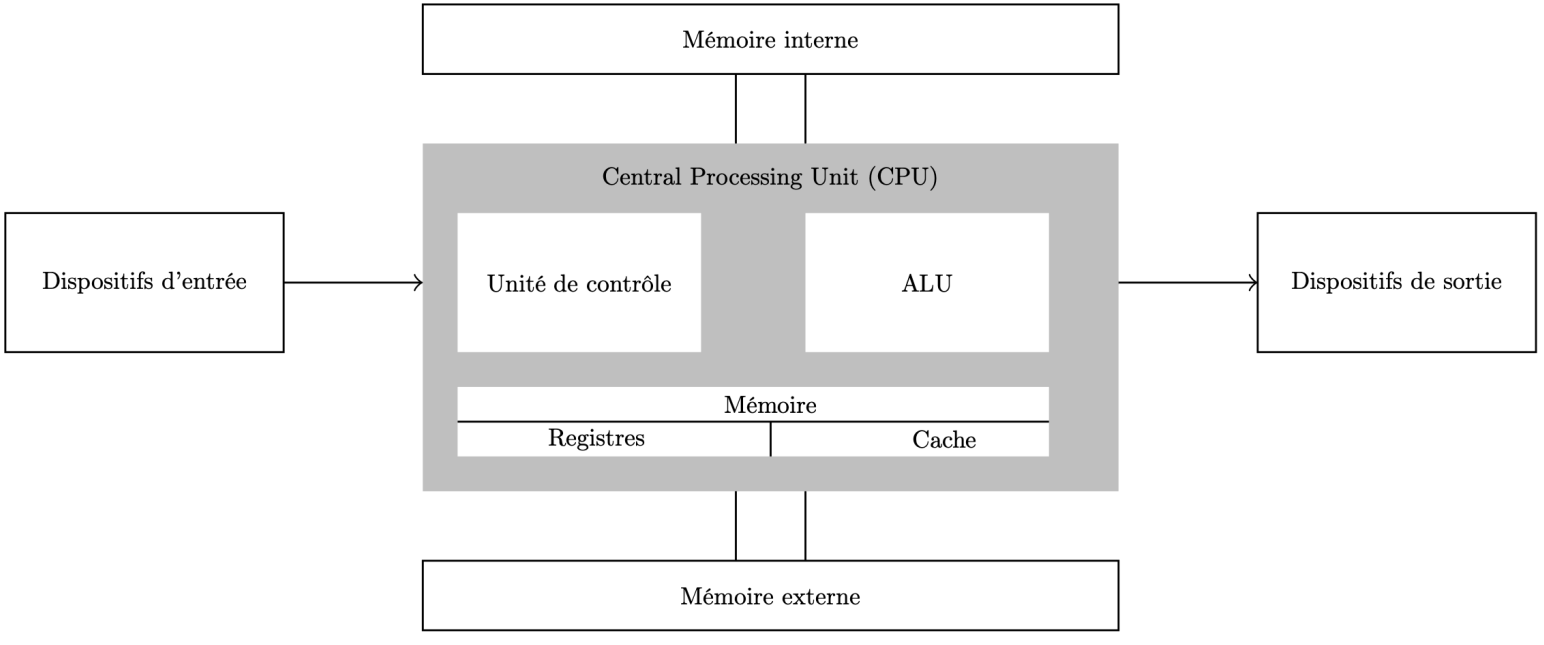
INSTITUT DE RECHERCHE EN INFORMATIQUE FONDAMENTALE

cnrs dépasser les frontières

- Tests et examens
  - CC : résultat des tests en TD / TP (semaine 4 et semaine 10/11)
  - E0 : partiel (samedi 26 octobre)
  - E1 : examen (**19 décembre de 8h30 à 11h:30**)
    - Examen écrit
  - E2 : examen fin juin
- Notes finales
  - Note session 1 : 25% CC + 25% E0 + 50% E1
  - Note session 2 : max(E2, 33% CC + 67% E2)
- Rappel
  - Pas de note $\Rightarrow$ pas de moyenne $\Rightarrow$ pas de semestre
- Site web
  - `moodlesupd.script.univ-paris-diderot.fr`

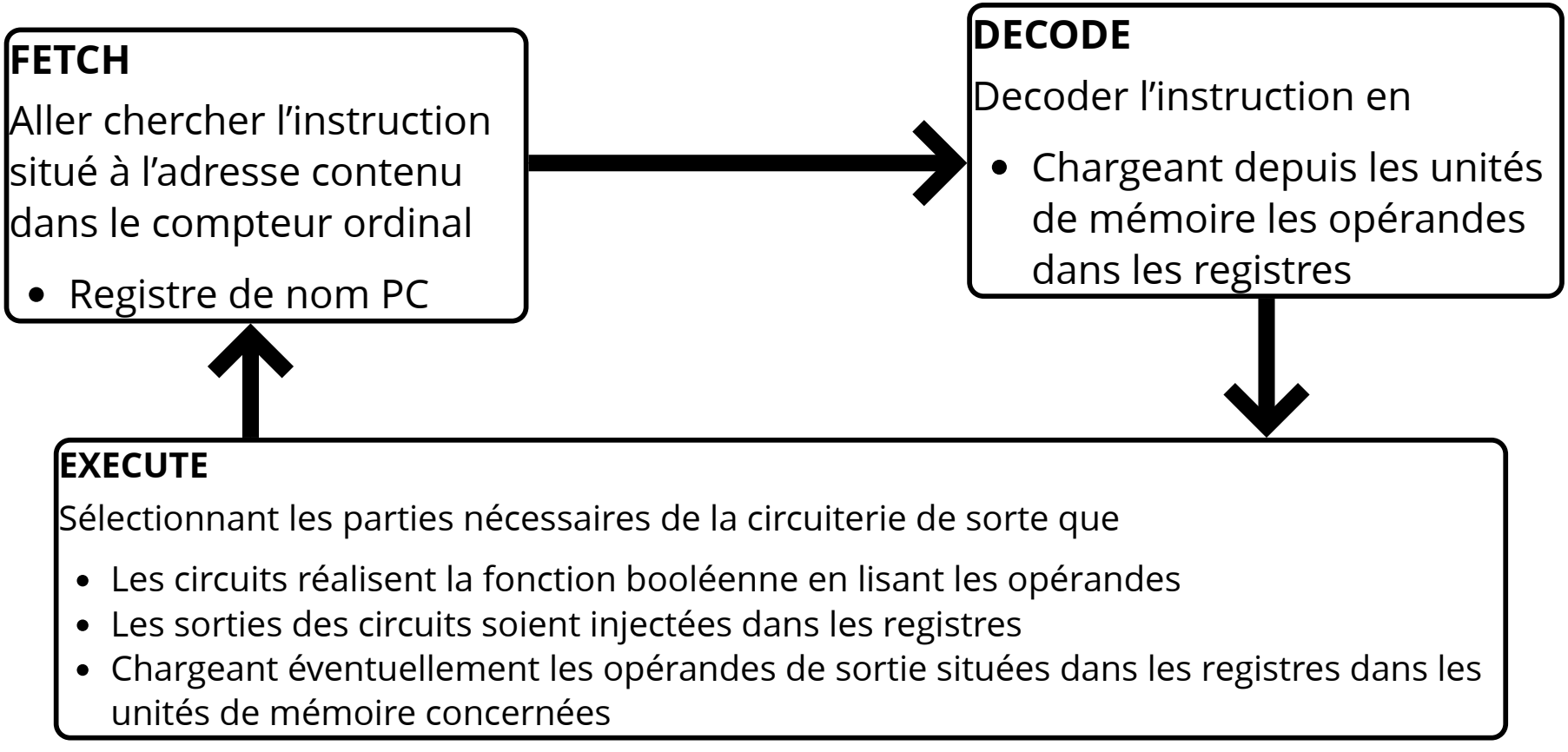
- Numération et arithmétique
- Numération et arithmétique en machine
- Numérisation et codage (texte, images)
- Compression, cryptographie, contrôle d'erreur
- Logique et calcul propositionnel
- **Circuits numériques**

- De quoi sont constitués (principalement pour ce qui nous concerne) les ordinateurs ?
  - D'un processeur effectuant des opérations élémentaires (CPU)
  - D'une mémoire interne
    - Les deux s'échangeant des informations continument
  - D'une mémoire externe et de dispositifs d'entrée et de sortie



- Les ordinateurs sont des circuits complexes mais dont le principe de fonctionnement est assez simple
  - Le cœur est le processeur dont le rôle est d'exécuter des instructions en provenance de la mémoire
  - Une instruction est lue depuis la mémoire et interprétée en
    - Lisant des valeurs depuis la mémoire
    - Sélectionnant la partie du circuit permettant de réaliser la fonction booléenne correspondante
    - Écrivant le résultat en mémoire
  - Les instructions sont codées
    - Le code permet par l'intermédiaire d'un décodeur de sélectionner le circuit qui réalisera le calcul
    - Les codes sont des mots binaires
      - Par exemple pour un processeur de la famille Little Computer 3, la négation du registre R1 se code sur 16 bits par 1001001001111111

- L'exécution d'une instruction consiste en les opération



- Passer à l'instruction suivante (généralement passage en séquence, mais parfois saut "if-then")

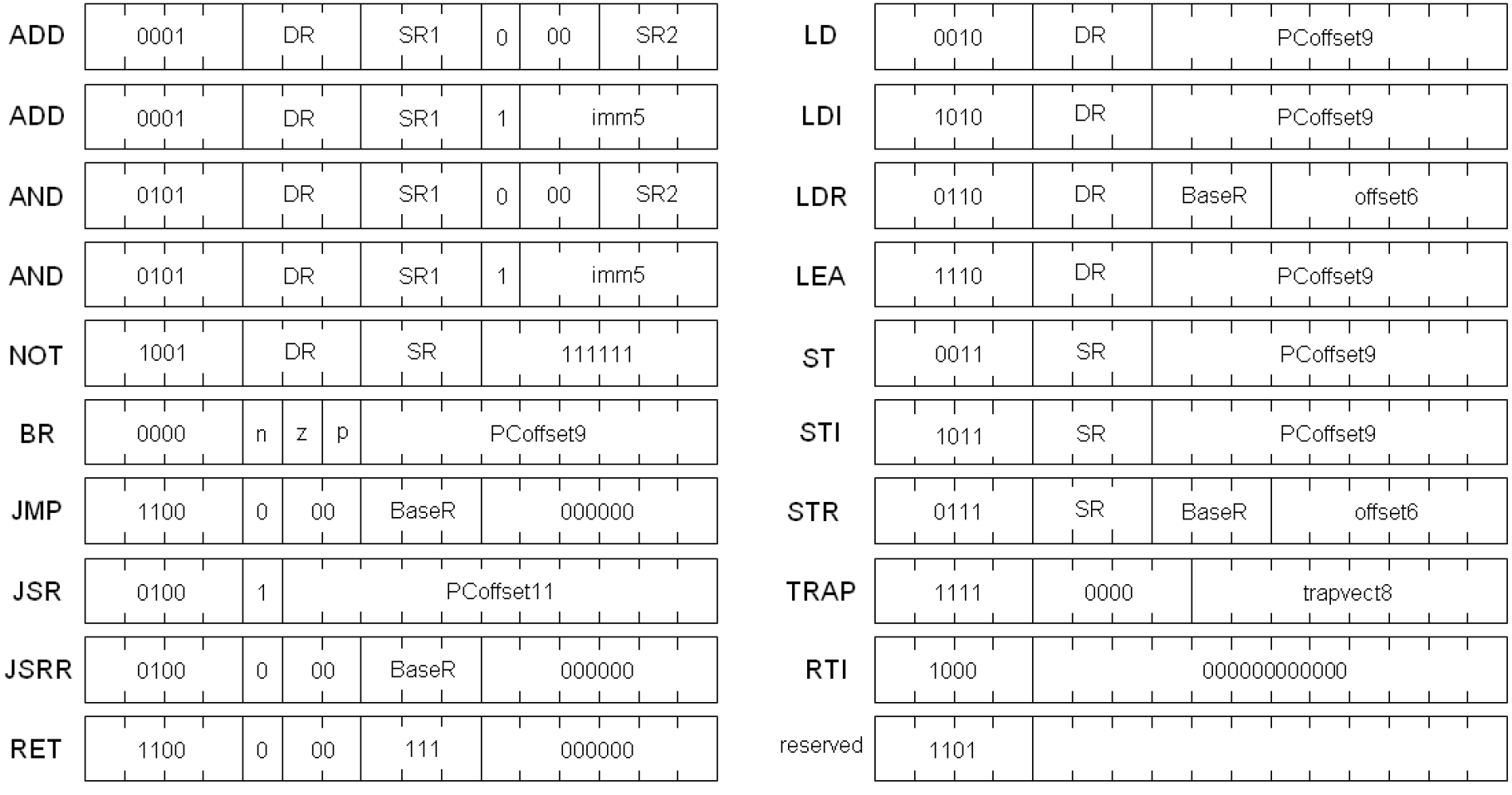
- Une machine à mots de 16 bits avec 8 registres de 16 bits et une mémoire vive (RAM)
  - Espace d'adressage de 2^{16} mots
  - Les registres sont nommés R0-R7
  - Une instruction est toujours codée sur 16 bits (un mot)
  - Arithmétique signée en complément a deux (pas de flottants)
  - La machine possède un registre d'état (CC)
    - Des indicateurs positionnés par certaines instructions lorsque certaines conditions sont rencontrées
      - N (valeur négative produite)
      - P (valeur positive produite)
      - Z (valeur nulle produite)
  - La machine possède un pointeur d'instruction (PC) contenant à tout instant l'adresse de la prochaine instruction à exécuter

- Mon premier programme
  - Additionne deux nombres en mémoire

```
            .ORIG x3000
            LD R0,memi
            LD R1,memj
            ADD R2,R0,R1
            ST R2,memk
            HALT
memi        .FILL #23
memj        .FILL #78
memk        .BLKW #1
            .END
```

- L'assembleur permet d'utiliser des symboles pour désigner des adresses mémoire (données ou instructions)
- L'assembleur fournit des pseudos-instructions permettant de déclarer des données, des adresses spéciales, etc
  - .ORIG : adresse de chargement en machine du programme
    - .ORIG x3000 : pseudo-instruction permettant de charger le programme assemblé à partir de l'adresse 0x3000
  - .FILL : adresse d'un mot de 16 bits (variable)
    - vari .FILL x1234 : déclaration de mot mémoire (16 bits), initialisé aux valeur 0x1234
  - .BLKW : adresse d'un groupe de mots de 16 bits
    - zone .BLKW #25 : déclaration d'une zone de mémoire de 25 mots
  - .STRINGZ : adresse d'une chaîne de caractères
    - hello .STRINGZ "Bonjour" : permet de déclarer une zone initialisée avec les caractères de la chaîne
    - Un caractère NUL est inséré à la fin pour marquer celle-ci
    - Un caractère par mot de 16-bits
  - .END : fin du code source du programme assembleur

- Instructions



- Multiplication par 10
  - $R0 \leftarrow 10 \times R1$

```
.ORIG   x3000
ADD     R0,R1,R1
ADD     R0,R0,R0
ADD     R0,R0,R1
ADD     R0,R0,R0
HALT
.END
```

- x-ou
  - $R3 \leftarrow R1 \oplus R2$

```
.ORIG   x3000
NOT     R1,R1
AND     R3,R1,R2
NOT     R1,R1
NOT     R2,R2
AND     R4,R1,R2
NOT     R2,R2
NOT     R3,R3
NOT     R4,R4
AND     R3,R3,R4
NOT     R3,R3
HALT
.END
```

- Nombre de bits égal à 1
  - $R0 \leftarrow$ nombre de bits égal à 1 dans $R1$

```
        .ORIG   x3000
        AND     R0,R0,#0
        ADD     R1,R1,#0
        BRzp    skipf
        ADD     R0,R0,#1
skipf   AND     R2,R2,#0
        ADD     R2,R2,#15
loop    ADD     R1,R1,R1
        BRzp    skip
        ADD     R0,R0,#1
skip    ADD     R2,R2,#-1
        BRp     loop
        HALT
        .END
```