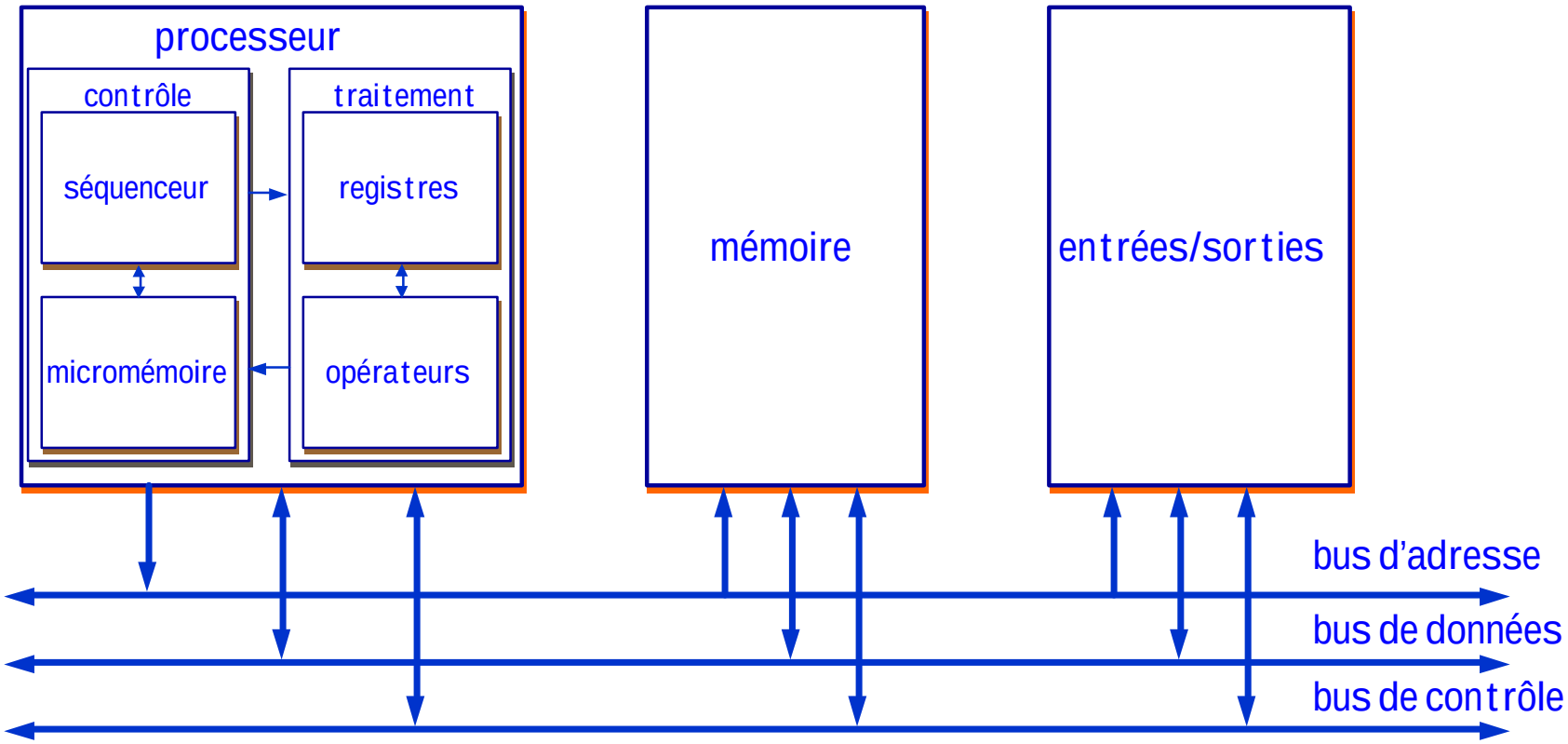
Types et performances des processeurs

Eduardo Sanchez
Laboratoire de Systèmes Logiques
Ecole Polytechnique Fédérale de Lausanne

Structure d'un ordinateur

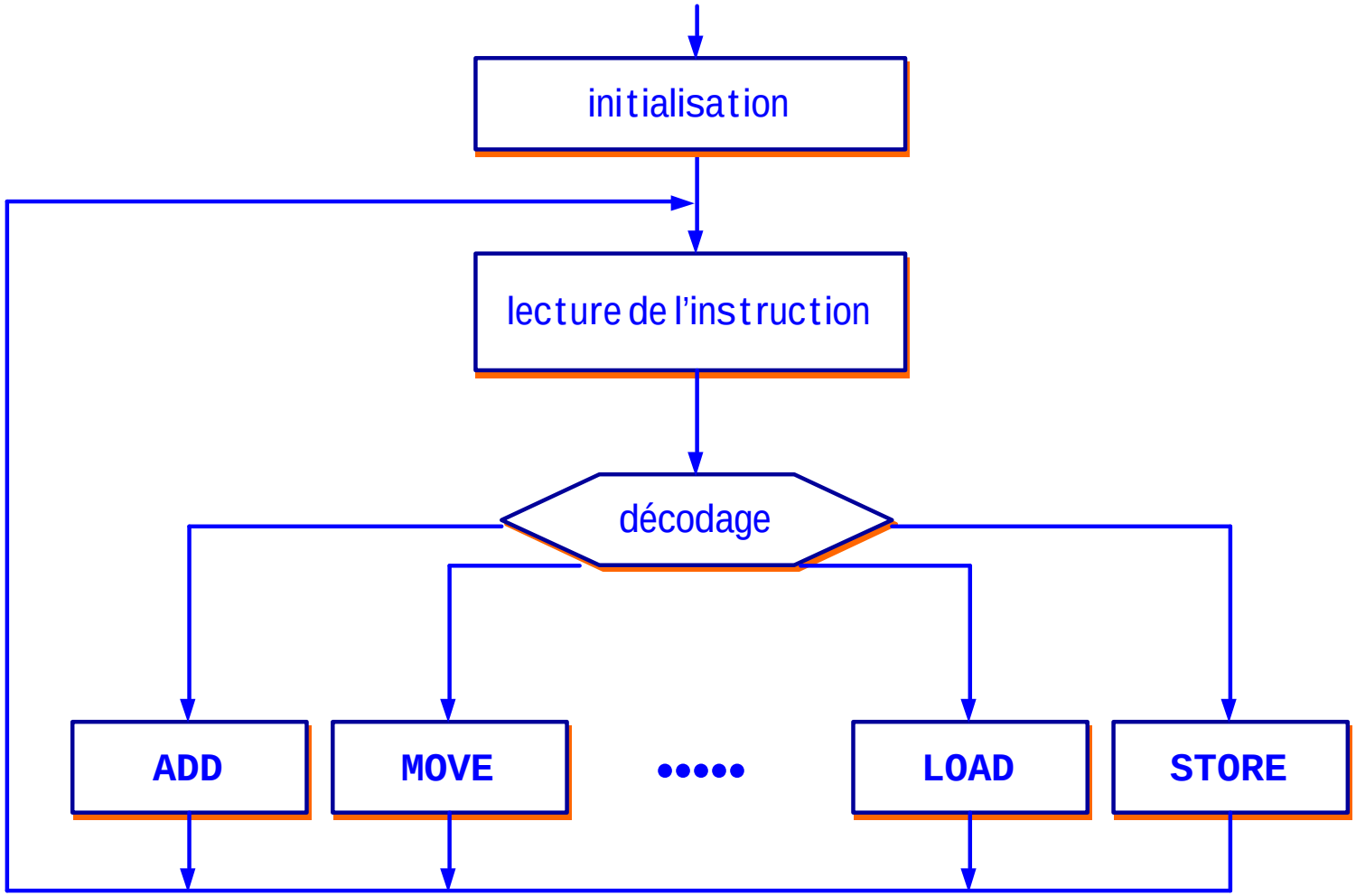


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

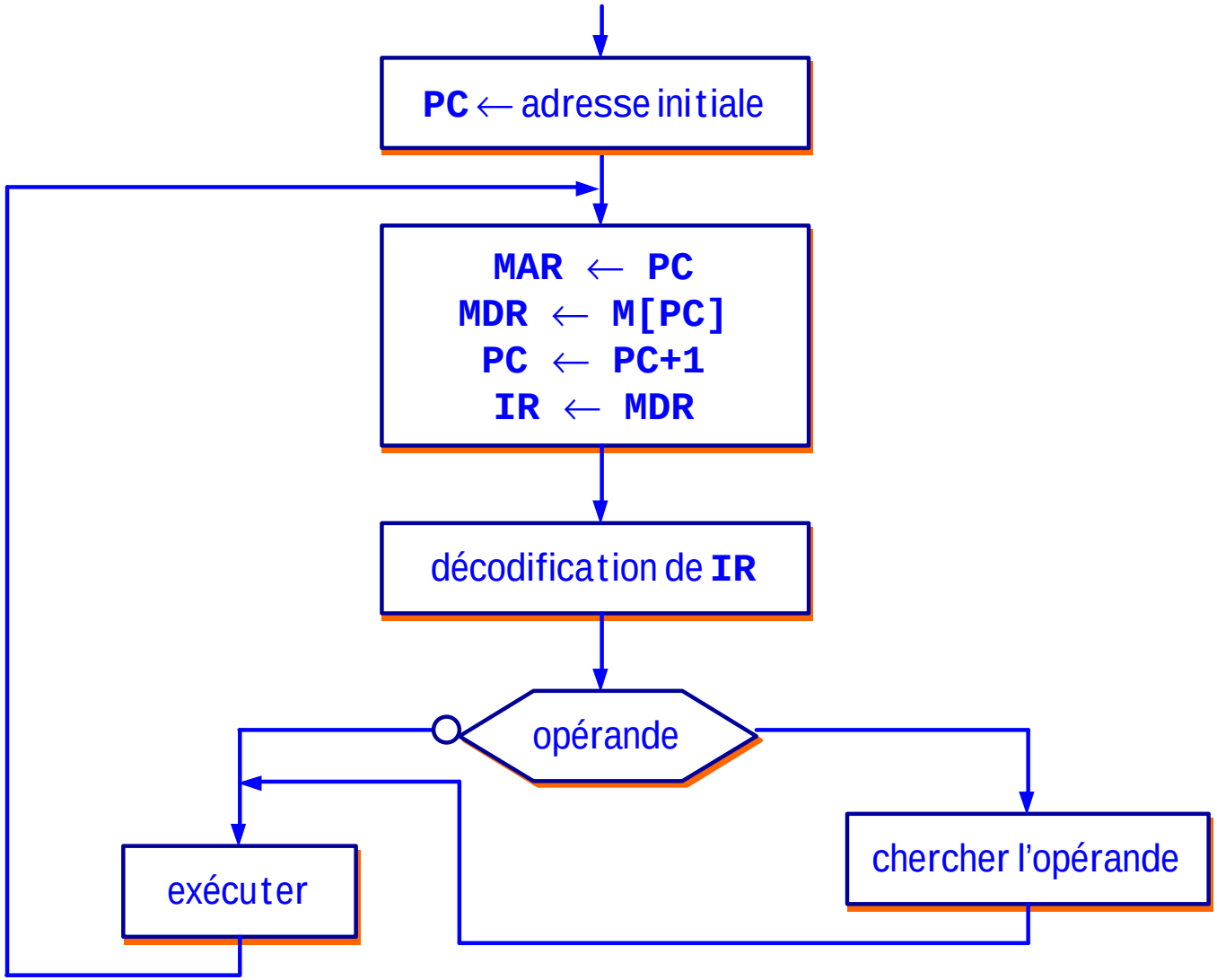
Exécution d'une instruction

- Un processeur effectue sans arrêt une boucle composée de trois phases:
  - recherche (*fetch*) de l'instruction: l'adresse en mémoire de l'instruction à exécuter est stockée en permanence dans un registre du processeur, appelé **PC** (*Program Counter*). L'instruction pointée par le **PC** est cherchée dans la mémoire et stockée dans un autre registre du processeur: le **IR** (*Instruction Register*)
  - décodage de l'instruction (*decode*): chaque instruction est identifiée, grâce à un code (*opcode*). En fonction de ce code, le processeur choisit la tâche à exécuter, c'est-à-dire la séquence de micro-instructions à exécuter
  - exécution (*execute*) de l'instruction: à la fin de cette phase, on retourne à la première phase

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne



Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne



Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Instructions machine

- Chaque instruction machine (assembleur) doit avoir les éléments suivants:
  - le code de l'opération (*opcode*): un code binaire identifiant l'opération à réaliser (addition, décalage, etc)
  - la référence de l'opérande source: il peut y en avoir plusieurs
  - la référence de l'opérande destination
  - la référence à la prochaine instruction à exécuter: une indication d'où chercher la prochaine instruction. Dans la plupart de cas, cette référence est inutile, implicite
- Les opérandes, source et destination, peuvent être cherchés à trois endroits différents:
  - la mémoire principale
  - les registres internes
  - les dispositifs d'entrée/sortie (périphériques)

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Représentation des instructions:
  Chaque instruction est stockée dans la mémoire comme une chaîne de bits. L'instruction est divisée en plusieurs champs, correspondant à ses différents éléments. L'organisation de ces différents champs est appelée le format de l'instruction.
  Il est possible, pour un même processeur, d'avoir plusieurs formats, de longueurs différentes.
  Pour faciliter la lecture des programmes en langages machine, on utilise une représentation symbolique, où les opcodes sont représentés par des abréviations appélées mnémoniques. Des exemples courants sont:
  - **ADD** addition
  - **SUB** soustraction
  - **LOAD** chargement d'un registre interne à partir de la mémoire
  - **STORE** chargement de la mémoire à partir d'un registre interne

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Types d'instruction:

  Un processeur doit avoir un ensemble d'instructions machine (le répertoire d'instructions) qui lui permettent de réaliser n'importe quel traitement d'information. Vu d'une autre façon, le répertoire d'instructions d'un processeur doit être capable d'interpréter n'importe quelle instruction d'un langage de haut niveau.

  En général, on peut diviser les instructions d'un processeur en 4 classes:

  - traitement des données: instructions arithmétiques et logiques
  - gestion de la mémoire
  - mouvement des données
  - contrôle: instructions de saut dans le programme

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Nombre d'opérandes:
  Le nombre d'opérandes, ainsi que leur emplacement, a une très grande importance, à cause de l'influence sur la taille des instructions et sur la vitesse d'exécution.
  Il est courant de trouver une classification des processeurs selon ce paramètre:
  - processeurs à accumulateur
  - processeurs à registres généraux
  - processeurs à pile
- Types d'opérandes:
  Les principaux types de données traités directement par les processeurs sont: adresses, nombres, caractères et données logiques
- Modes d'adressage:
  Cést la façon de spécifier l'adresse des opérandes

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Processeurs à accumulateur

- Les résultats de toutes les opérations sont stockés dans un registre particulier, l'accumulateur
- Toutes les variables sont stockées dans la mémoire
- Exemples: DEC PDP-8, Intel 8080, Motorola 6800
- Exemple de programme:

x := y + z

LOAD	**y**	**ACC** ← **M[y]**
ADD	**z**	**ACC** ← **ACC + M[z]**
STORE	**x**	**M[x]** ← **ACC**

- Le format d'une instruction doit utiliser deux champs:
  - le code de l'opération (*opcode*)
  - l'adresse de l'opérande (l'accumulateur est toujours un opérande par défaut)

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Processeurs à registres généraux

- Les variables utilisées le plus fréquemment sont stockées dans un ensemble de registres internes:
  - les accès sont plus rapides
  - les adresses sont plus courtes
- Exemples: IBM 360/370, DEC PDP-11, Intel x86, Motorola 68000, Sparc, PowerPC, MIPS
- Les opérations peuvent avoir lieu seulement avec les registres (architecture **LOAD/STORE**), ou avec un, deux ou trois opérandes en mémoire. Exemples:

```
ADD    Rx, Ry          Rx ← Rx + Ry
ADD    Rx, Ry, Rz      Rx ← Rx + Ry + Rz
ADD    Rx, y           Rx ← Rx + M[y]
ADD    x, y            M[x] ← M[x] + M[y]
ADD    x, y, z         M[x] ← M[y] + M[z]
```


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Processeurs à pile

- Les opérandes se trouvent toujours au sommet d'une pile, dans la mémoire. Et le résultat est toujours stocké au sommet de la pile. En conséquence, une instruction typique ne contient pas d'adresse d'opérande
- Un registre particulier du processeur, le **stack pointer** (**SP**), pointe toujours au sommet de la pile
- Exemples: Burroughs B5000, HP 300
- Les instructions de base sont:

PUSH	**x**	**M[SP]** $\leftarrow$ **M[x]**
POP	**x**	**M[x]** $\leftarrow$ **M[SP]**

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Exemple de compilation

- Phrase en langage de haut niveau:

```
P := (Q*R + S*T)*(U + V)
```

- Compilation pour un processeur à registres:

```
LD      R0, Q
MUL     R0, R
LD      R1, S
MUL     R1, T
ADD     R0, R1
LD      R1, U
ADD     R1, V
MUL     R0, R1
ST      R0, P
```


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Compilation pour un processeur à pile:

```
PUSH    Q
PUSH    R
MUL
PUSH    S
PUSH    T
MUL
ADD
PUSH    U
PUSH    V
ADD
MUL
POP     P
```


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Performance d'un processeur

- Deux paramètres peuvent être utilisés pour mesurer la performance d'un processeur:
  - le temps de réponse ou temps d'exécution d'une certaine tâche: temps écoulé entre le début et la fin d'exécution de la tâche
  - *throughput*: quantité total de travail réalisé dans un certain temps
- L'amélioration du temps de réponse implique toujours une amélioration du *throughput*. Toutefois, le contraire n'est pas toujours vrai: une augmentation du nombre de processeurs d'un ordinateur augmente le *throughput*, sans améliorer nécessairement le temps de réponse
- Nous allons considérer le temps d'exécution comme paramètre principal pour le calcul de la performance d'un processeur

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

$$\text{performance} = \frac{1}{\text{temps d'exécution}}$$

- Le rapport de performance entre deux machines A et B est:

$$\frac{\text{performance A}}{\text{performance B}} = n = \frac{\text{temps B}}{\text{temps A}}$$

  et l'on dit que A est n fois plus rapide que B

- Pour le calcul de performance, on tient compte seulement du *CPU time*, inférieur à l'*elapsed time*

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Le temps d'exécution dépend de trois facteurs:
  - le nombre d'instructions machine exécutées,
  - le nombre moyen de cycles d'horloge par instruction machine et
  - la période d'horloge

Temps = 1/ performance =
(nombre d'instructions) x
(nombre de cycles par instruction) x
(période d'horloge)

$$\text{Temps} = \frac{\text{IC} \times \text{CPI}}{\text{f}}$$

Instruction count

clock cycles per instruction

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Pour une architecture donnée (un certain répertoire d'instuctions), il est possible d'améliorer la performance par trois moyens différents:
  - augmenter la fréquence d'horloge
  - améliorer l'organisation interne pour diminuer le CPI
  - améliorer le compilateur pour diminuer le IC ou pour augmenter le taux d'utilisation des instructions avec un CPI moindre

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Processeurs RISC et CISC

- Lors de la conception d'une architecture, il est possible de privilégier l'un ou l'autre des paramètres qui interviennent dans la performance d'un processeur
- Il existe deux types de processeurs, selon le paramètre optimisé:
  - processeurs CISC (*Complex Instruction Set Computer*)
  - processeurs RISC (*Reduced Instruction Set Computer*)

Temps = 1/ performance =
(nombre d'instructions) x
(nombre de cycles par instruction) x
(période d'horloge)

CISC

RISC

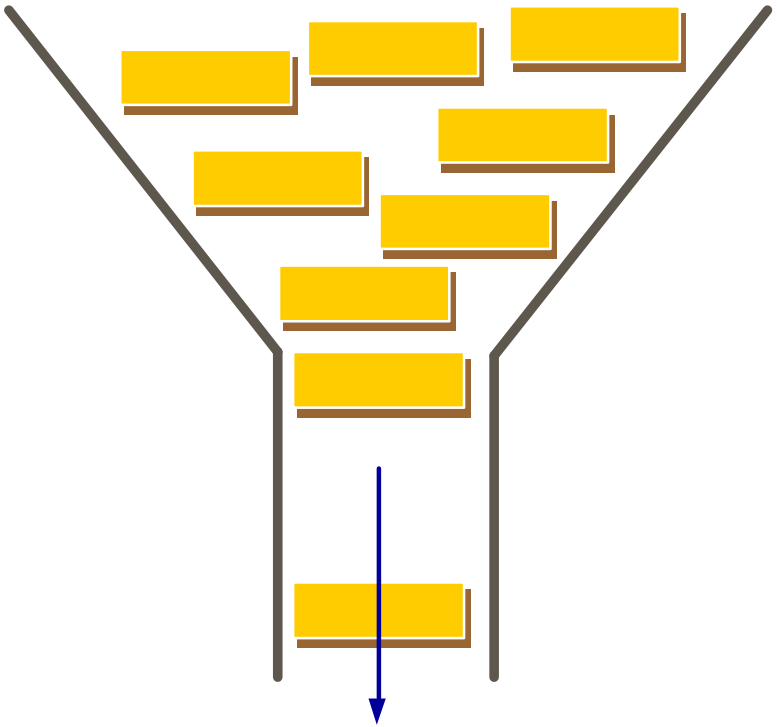
Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Caractéristiques des processeurs RISC

- Une seule taille d'instruction: 32 bits
- Très peu de modes d'adressage (absence d'adressage indirect)
- Architecture *load/store*: aucune opération avec la mémoire
- Nombre élevé de registres
- Pipeline
- Architectures superscalaires
- Mémoire cache

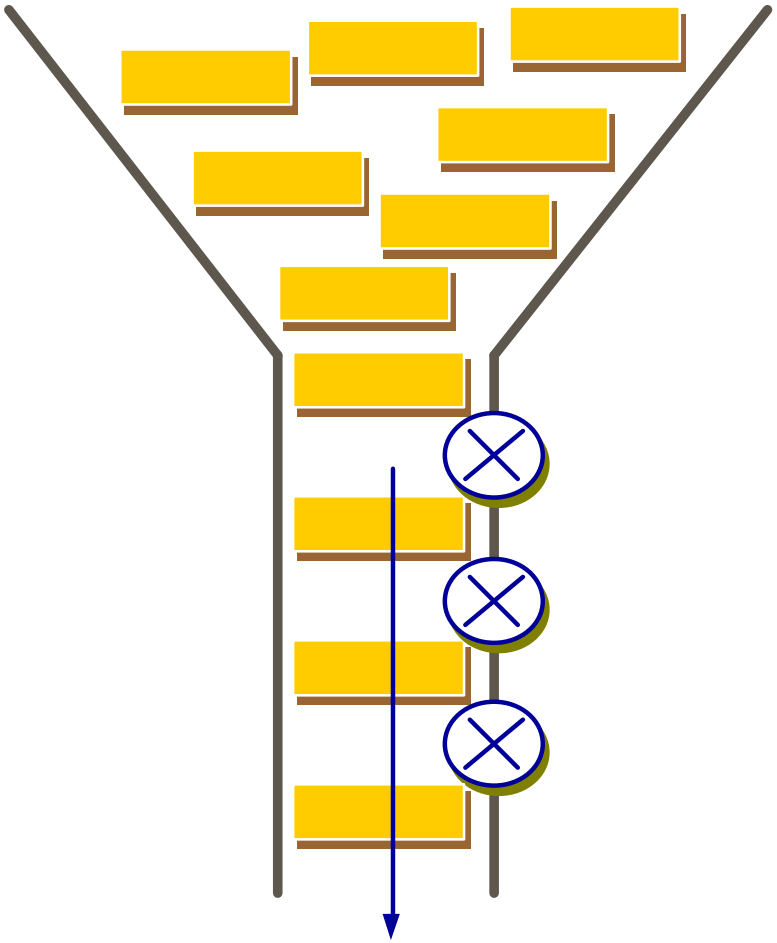
Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Architecture standard



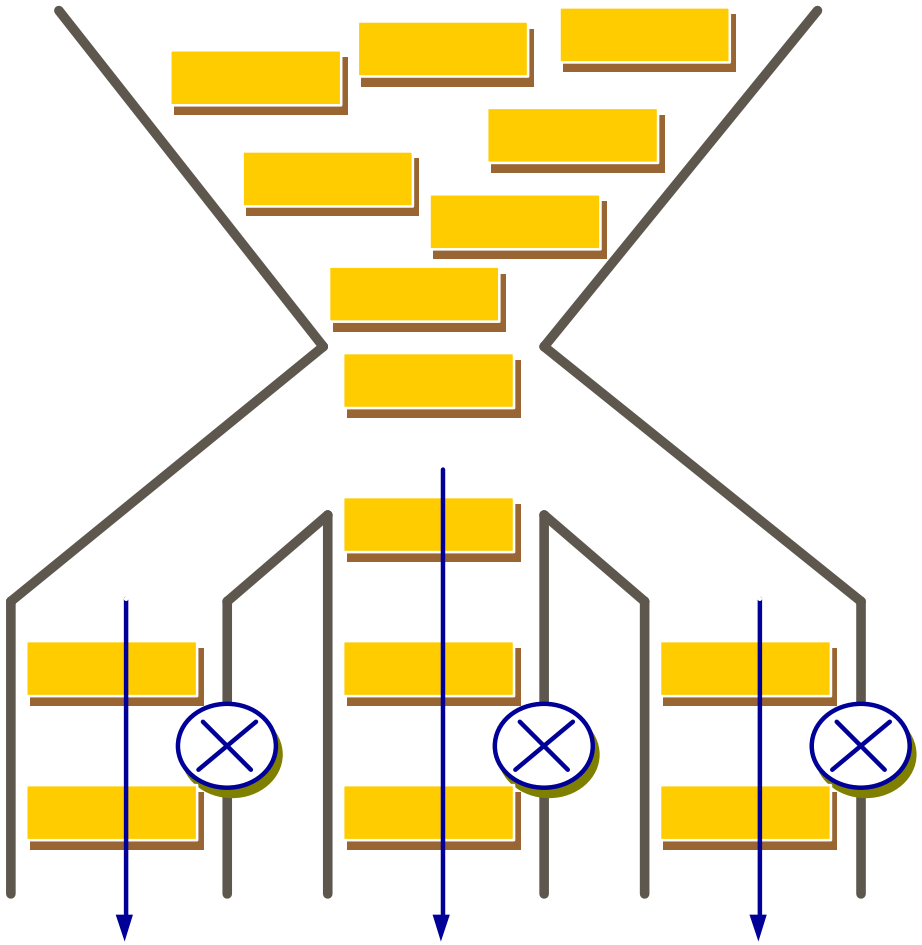
Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Le pipeline



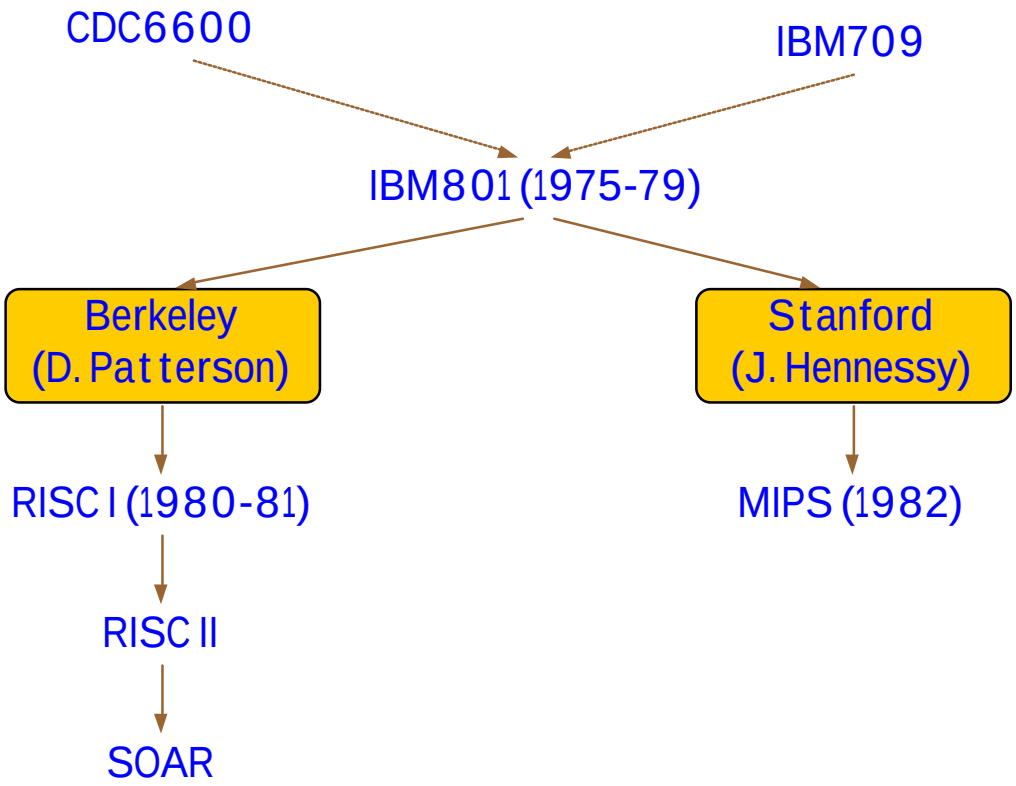
Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Architecture superscalaire



Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Histoire de RISC



Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Processeurs RISC commerciaux

- Silicon Graphics: MIPS — 1986, pipeline
- Hewlett-Packard: HP-PA — 1986, pipeline
- Sun: SPARC — 1987, superscalaire
- Apple-IBM-Motorola: Power PC — 1990, superscalaire
- DEC: Alpha — 1992, pipeline

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Mesure de performance: SPEC2000

- Pour résoudre le problème du choix des programmes de test (*benchmarks*) à utiliser pour mesurer la performance, plusieurs fabricants se sont associés pour créer SPEC (*Standard Performance Evaluation Corporation*)
- Il y deux groupes principaux de *benchmarks*, l'un pour les nombres entiers et l'autre pour les nombres réels. Une valeur de performance est calculée pour chacun de ces groupes, appelée SPECint et SPECfp. Une seule valeur est donnée pour chaque groupe: la moyenne géométrique
- La valeur *baseline* est calculée avec un maximum de 4 flags de compilation, les mêmes pour tous les programmes

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Le premier groupe de *benchmarks* a été publié en 1989, avec des nouvelles versions en 1992, 1995 et 2000
- SPEC CPUint2000: 12 programmes écrits en C et C++
- SPEC CPUfp2000 : 14 programmes écrits en Fortran et C
- La machine de référence est une Sun Ultra5_10 à 300 MHz: sa performace reçoit une valeur de 100 SPEC2000
- La performance mesurée par SPEC ne dépend pas seulement du processeur: elle dépend également du système de mémoire et du compilateur. Par contre, elle ne tient pas compte des entrées/sorties, du réseau ni du graphisme
- Le site web de SPEC est: http://www.spec.org

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Benchmark	Language	KLOC	Resident size (Mbytes)	Virtual size (Mbytes)	Description
SPECint2000					
164.gzip	C	7.6	181	200	Compression
175.vpr	C	13.6	50	55.2	FPGA circuit placement and routing
176.gcc	C	193.0	155	158	C programming language compiler
181.mcf	C	1.9	190	192	Combinatorial optimization
186.crafty	C	20.7	2.1	4.2	Game playing: Chess
197.parser	C	10.3	37	62.5	Word processing
252.eon	C++	34.2	0.7	3.3	Computer visualization
253.perlbmk	C	79.2	146	159	Perl programming language
254.gap	C	62.5	193	196	Group theory, interpreter
255.vortex	C	54.3	72	81	Object-oriented database
256.bzip2	C	3.9	185	200	Compression
300.twolf	C	19.2	1.9	4.1	Place and route simulator
SPECfp2000					
168.wupwise	F77	1.8	176	177	Physics: Quantum chromodynamics
171.swim	F77	0.4	191	192	Shallow water modeling
172.mgrid	F77	0.5	56	56.7	Multigrid solver: 3D potential field
173.applu	F77	7.9	181	191	Partial differential equations
177.mesa	C	81.8	9.5	24.7	3D graphics library
178.galgel	F90	14.1	63	155	Computational fluid dynamics
179.art	C	1.2	3.7	5.9	Image recognition/neural networks
183.equake	C	1.2	49	51.1	Seismic wave propagation simulation
187.facerec	F90	2.4	16	18.5	Image processing: Face recognition
188.ammp	C	12.9	26	30	Computational chemistry
189.lucas	F90	2.8	142	143	Number theory/primality testing
191.fma3d	F90	59.8	103	105	Finite-element crash simulation
200.sixtrack	F77	47.1	26	59.8	Nuclear physics accelerator design
301.apsi	F77	6.4	191	192	Meteorology: Pollutant distribution

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

Processeurs actuels

	RISC					CISC	
	Alpha 21264B	IBM Power3-II	Sun Ultra-III	HP PA-8600	MIPS R12000	Intel P4	AMD Athlon
Fréquence (MHz)	833	450	900	552	400	1500	1200
Superscalaire	4	4	4	4	4	3	3
Pipeline	7/9	7/8	14/15	7/9	6	22/24	9/11
Désordre (instr)	80	32	0	56	48	126	72
Technologie	0.18-6	0.22-6	0.18-7	0.25-2	0.25-4	0.18-6	0.18-6
Taille (mm^2)	115	163	210	477	204	217	117
Transistors (M)	15.4	23	29	130	7.2	42	37
Puissance (W)	75	36	65	60	25	55	76
Cache (I/D)	64K/64K	32K/64K	32K/64K	512K/1M	32K/32K	12K/8K	64K/64K
SPECint2000b	518	286	438	417	320	524	443
SPECfp2000b	590	356	427	400	319	549	387

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne