

Conception de circuits numériques et architecture des ordinateurs

Équipe pédagogique :

Marie Badaroux (TD), Julie Dumas (TD+TP), Florence Maraninchi (TD+TP),
Olivier Muller (TD+TP), **Frédéric Pétrot** (CM+TD), Pierre Ravenel (TD+TP)
Eduardo Tomasi Ribeiro (TP), Lionel Rieg (TD+TP),
Sébastien Viardot (TD+TP Apprentis)



Année universitaire 2024-2025

- C1 Codage des nombres en base 2, logique booléenne, portes logiques, circuits combinatoires
- C2 Circuits séquentiels
- C3 Construction de circuits complexes
- C4 Micro-architecture et fonctionnement des mémoires
- C5 Machines à état
- C6 Synthèse de circuits PC/PO
- C7 Optimisation de circuits PC/PO
- C8 Interprétation d'instructions - 1
- C9 **Interprétation d'instructions - 2**
- C10 Interprétation d'instructions - 3
- C11 Introduction aux caches

Plan détaillé du cours d'aujourd'hui

1 Processeur et jeu d'instruction

- Introduction
- Relation ISA/Architecture
- Encodage des instructions

Plan

1 Processeur et jeu d'instruction

- Introduction
- Relation ISA/Architecture
- Encodage des instructions

Introduction

Rappel de l'épisode précédent

- Interprétation de données binaires considérées comme des instructions
- Lues implicitement et en séquence
- ISA défini les changement d'états de la machine qu'induit chaque instruction

ISA : exemples typiques

Usage : $f = a \times b - (a + c \times b)$

a, b, c variables en mémoire,

$\%r_i, \%r_j$ registres du processeur

stack HP3000, Sun picojava, ...
 accu mostek 6502, Motorola 68H12,
 Microchip PIC, ...
 2-addr Motorola 68k, Intel x86, ...
 3-addr MIPS rx00, Sun Sparc Vx,
 IBM Power, ARM Vx, ...

stack	accu	2-addr	3-addr	3-addr-ld/st
push a	clear	xor %r1, %r1	mult %r1, b, c	ld %r1, c
push b	add c	add %r1, b	add %r1, %r1, a	ld %r2, b
mult	mult b	mult %r1, c	mult %r2, a, b	mult %r1, %r1, %r2
push a	add a	add %r1, a	sub f, %r2, %r1	ld %r3, a
push c	st x	xor %r2, %r2		add %r1, %r1, %r3
push b	clear	add %r2, b		mult %r2, %r2, %r3
mult	add a	mult %r2, a		sub %r3, %r2, %r1
add	mult b	sub %r2, %r1		st %r3, f
sub	sub x	mov f, %r2		
pop f	st f			

Influence de l'ISA sur l'architecture : Machine à Pile

Instructions

push a	$push(mem[a])$
pop a	$mem[a] := pop$
cpush n	$push(mem[pc + 1])$
add	$push(pop + pop)$
sub	$push(pop - pop)$
mult	$push(pop \times pop)$

De plus, pour chaque instruction

Incrémentation de pc

$pc := pc + 1$

Chargement de l'instruction suivante

$ir := mem[pc]$

Influence de l'ISA sur l'architecture : Machine à Pile

Encodage des instructions

		push a	000
		pop a	001
		cpush n	010
		add	100
		sub	101
		mult	110
Instructions sur des octets			
7-5	4—0		
<hr/>			
OPCODE	00000		
<hr/>			

Pour push a et pop, les 2 octets suivants contiennent l'adresse

Pour cpush, l'octet suivant contient la constante

Ressources

Instructions sur 8 bits

Memoire programme et données de 65536 (2^{16}) octets

Principe de l'exécution

```

1  state := RESET;
2  while true do
3      switch state do
4          case RESET do
5              | state := FETCH, PC := @reset
6          case FETCH do
7              | state := DECODE, IR := MEM[PC], PC := PC + 1
8          case DECODE do
9              switch IR7..5 do
10                 case 00- do state := GETLADDR;
11                 case 010 do state := PUSHC;
12                 case 1-- do state := OP1;
13             end switch
14         case GETLADDR do
15             | state := GETHADDR, ADDR7..0 := MEM[PC], PC := PC + 1
16         case GETHADDR do
17             if IR5 = 0 then state := PUSH;
18             else state := POP;
19             ADDR15..7 := MEM[PC], PC := PC + 1
20         case ... do ...;
31     end switch
32 end while

```

Principe de l'exécution

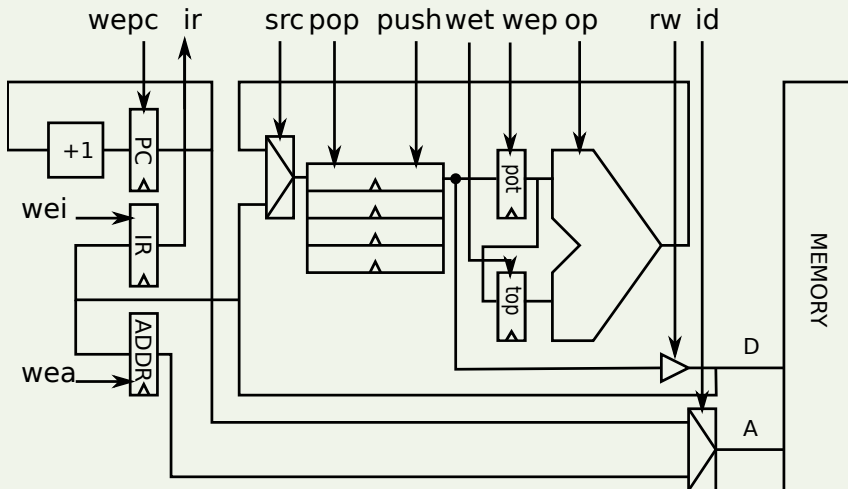
```

1  state := RESET;
2  while true do
3      switch state do
4          case ... do ...;
20         case PUSHC do state := FETCH, push(MEM[PC]), PC := PC + 1;
21         case OP1 do state := OP2, top := pop();
22         case OP2 do state := OP3, pot := pop();
23         case OP3 do
24             switch IR6..5 do
25                 case 00 do state := ADD;
26                 case 01 do state := SUB;
27                 case 10 do state := MULT;
28             end switch
29         case ADD do state := FETCH, push(pot + top);
30         case SUB do state := FETCH, push(pot - top);
31         case MULT do state := FETCH, push(pot × top);
32     end switch
33 end while

```

Influence de l'ISA sur l'architecture : Machine à Pile

Ressources de la Partie Opérative



Influence de l'ISA sur l'architecture : Machine à Accumulateur

Instructions

clear	$accu := 0$
add a	$accu := accu + mem[a]$
sub a	$accu := accu - mem[a]$
mult a	$accu := accu \times mem[a]$
st a	$mem[a] := accu$
add n	$accu := accu + ir_{3..0}$
sub n	$accu := accu - ir_{3..0}$
mult n	$accu := accu \times ir_{3..0}$

Incrémentation de pc	$pc := pc + 1$
Chargement de l'instruction suivante	$ir := mem[pc]$

Influence de l'ISA sur l'architecture : Machine à Accumulateur

Encodage des instructions

Instructions sur des octets

7-4	3-0
OPCODE	nnnn

clear 0000

add a 1000

mult a 1001

sub a 1010

st a 1011

add n 1100

mult n 1101

sub n 1110

Pour opérations mémoire, 2 octets suivants contiennent adresse

Pour opérations avec constante, $-8 \leq n \leq +7$

Ressources

Instructions sur 8 bits

Mémoire programme et données de 65536 (2^{16}) octets

Principe de l'exécution

```
1  state := RESET;
2  while true do
3      switch state do
4          case RESET do state := FETCH, PC := @reset;
5          case FETCH do state := DECODE, IR := MEM[PC], PC := PC + 1;
6          case DECODE do
7              switch IR7..4 do
8                  case 0000 do state := CLEAR;
9                  case 10- do state := GETLADDR;
10                 case 1100 do state := ADDC;
11                 case 1101 do state := MULTC;
12                 case 1110 do state := SUBC;
13             end switch
14         case GETLADDR do
15             state := GETHADDR, ADDR7..0 := MEM[PC], PC := PC + 1
16         case ... do ...;
17     end switch
18 end while
```

Principe de l'exécution

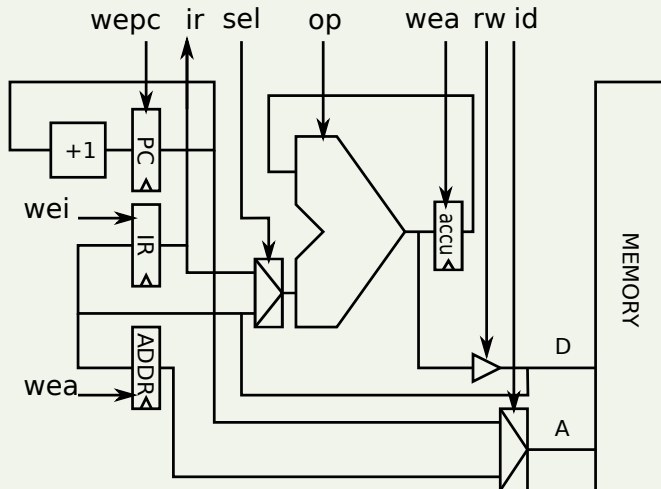
```

1  state := RESET;
2  while true do
3      switch state do
4          case ... do ...;
16     case GETHADDR do
17         switch IR5..4 do
18             case 00 do state := ADD;
19             case 01 do state := MULT;
20             case 10 do state := SUB;
21         end switch
22         ADDR15..7 := MEM[PC], PC := PC + 1
23     case ADD do state := FETCH, accu := accu + MEM[ADDR];
24     case MULT do state := FETCH, accu := accu × MEM[ADDR];
25     case SUB do state := FETCH, accu := accu − MEM[ADDR];
26     case ST do state := FETCH, MEM[ADDR] := accu;
27     case ADDC do state := FETCH, accu := accu + IR34 || IR3..0;
28     case MULTC do state := FETCH, accu := accu × IR34 || IR3..0;
29     case SUBC do state := FETCH, accu := accu − IR34 || IR3..0;
30 end switch
31 end while

```

Influence de l'ISA sur l'architecture : Machine à Accumulateur

Ressources de la Partie Opérative



Encodage des instructions

Dépend :

- du nombre d'opérations
- du nombre d'opérandes
- du nombre de registres
- des modes d'adressage

Peut être :

- de taille variable
les instructions peuvent aller de 1 à 13 octets par exemple (x86)
et possèdent des *formats* différents
- de taille fixe
toutes les instructions ont la même taille, *i.e.* 32 bits (ARM)
mais ne possèdent pas pour autant le même *format*

Encodage des instructions

2-adresses, 4 registres, instruction sur 8 bits

2 bits par registre

MSB à 0, opérations entre 2 registres

MSB à 1, opérations entre 1 registre et la mémoire

opérande stocké après l'instruction (1 ou 2 octets)

Exemples :

add %r1, %r0

add %r2, \$12

add %r3, addr

formats d'instruction

Regs :	7-4	3-2	1-0	Const :	7-4	3-2	1-0	7 - 0
	OPCODE	Rs	Rd		OPCODE	00	Rd	CSTE

Mém :	7-4	3-2	1-0	7 - 0	7 - 0
	OPCODE	00	Rd	ADDRH	ADDRH

Résumé

Implantation des processeurs

- Choix d'un ISA
- Choix de formats d'instructions adaptés
- Implantation « naturelle » sous la forme Partie Contrôle/Partie Opérative