

Outline

- Représentation des nombres
- Circuits logiques
- Unité Arithmétique et Logique
- Notions de temps et de mémorisation
- Contrôle et jonction des composants
- Evolution des ordinateurs – Historique
- **A simple microprocessor**
- Programmation d'un microprocesseur
- Système complet
- Les microprocesseurs actuels
- Exploitation de la performance des microprocesseurs

RISC Processors

- Architectures and complex instruction sets:
 - ⇒ Long design time (control circuit,...)
 - ⇒ Fast evolution of technology not fully taken advantage of
 - ⇒ Operands in memory → long access time
 - ⇒ More complex compilers
- RISC (*Reduced Instruction Set Computer*):
 - Few instructions, few addressing modes, few data formats, fixed instruction size.
 - Operands in registers only for fast access.
 - Instructions decomposed into pipeline stages
 - ⇒ More simple design, low cycle time.

Instruction Sets

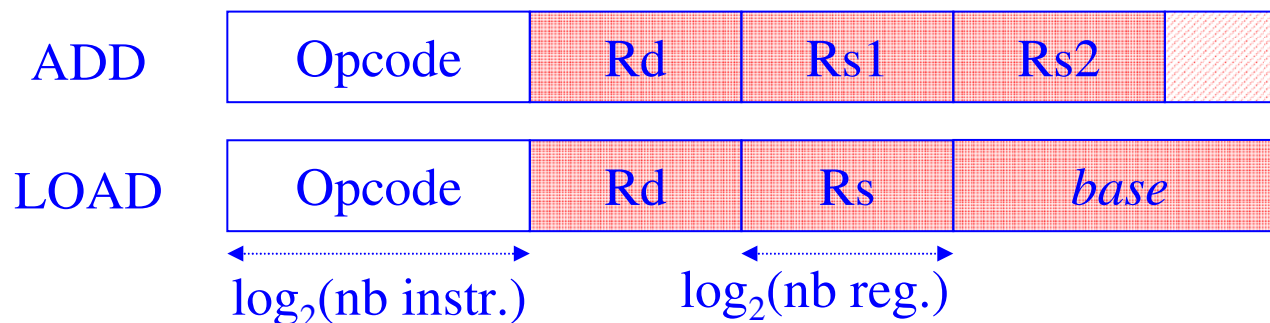
- The instruction set describes an abstract version of a processor (*ISA: Instruction Set Architecture*).
- An instruction set can correspond to many different implementations.
 - Example: x86 ISA and Intel processors.
- The instruction set must be able to:
 - Access memory
 - Perform arithmetic and logic operations
 - Control the program flow (branching)

Designing a Simple Processor

« Specifications » of the LC-2

- Use a 16-bit word:
 - Circuit size and speed constraints
 - Memory size
- A RISC processor (load/store, register operands, fixed-size instructions)
- Minimum ISA:
 - ALU: addition and minimum number of logic operations; operands: either registers, or immediate;
 - Memory: absolute addressing, index addressing
 - Control: unconditional branches, procedure calls, conditional branches (possible tests: zero, positive, negative); system calls; direct and indexed addressing
- Questions:
 - What ISA format ?
 - What implementation for the processor ?

Instruction Set



- Instructions size = balance between
 - Number of instructions/operands
 - Number of registers
 - Memory size
 - Control circuit complexity
- Example: 16 bits for data word/instructions:
 - 16 instructions \rightarrow 4 opcode bits.
 - 12 bits for operands (3 registers for ALU instructions; example: ADD $Rd \leftarrow Rs1, Rs2$)
 - \Rightarrow maximum 4 bits per register number (16 registers)
 - *base* size (LOAD $Rd \leftarrow Rs1, base$)
 - \Rightarrow depends on register size (4 bits for 16 registers, 6 bits for 8 registers)
- Predict the most important characteristics of upcoming architectures:
 - Adding new instructions
 - Increasing the number of registers
 - Facilitate memory access (large base size)
 - ...

Instruction Set Design

- Arithmetic and logic instructions:

- Be able to write any logic expression.
- Be able to make any arithmetic computation.
- Two addressing modes:
 - ❖ Immediate
 - ❖ Direct

- ADD $Rd \leftarrow Rs1, Rs2$
 - ❖ $Rd \leftarrow Rs1 + Rs2$
- AND $Rd \leftarrow Rs1, Rs2$
 - ❖ $Rd \leftarrow ET(Rs1, Rs2)$
- ADD $Rd \leftarrow Rs, valeur$
 - ❖ $Rd \leftarrow Rs + valeur$
- AND $Rd \leftarrow Rs, valeur$
 - ❖ $Rd \leftarrow ET(Rs, valeur)$
- NOT $Rd \leftarrow Rs$
 - ❖ $Rd \leftarrow NOT(Rs)$

Instruction Set Design

- Memory access:

- Load
- Store
- 3 addressing modes:
 - ❖ Direct
 - ❖ Indexed
 - ❖ Immediate

- LD $Rd \leftarrow M(\text{adresse})$
- ST $Rs \rightarrow M(\text{adresse})$
 - ❖ Direct addressing
 - ❖ Address = $PC[15:9], \text{offset}[8:0]$
- LDR $Rd \leftarrow Rs, \text{index}$
 - ❖ Direct addressing
 - ❖ Address = $Rs + \text{index}$
- STR $Rs2 \rightarrow Rs1, \text{index}$
 - ❖ Direct addressing
 - ❖ Address = $Rs1 + \text{index}$
- LEA $Rd \leftarrow \text{adresse}$
 - ❖ Direct addressing
 - ❖ $Rd \leftarrow PC[15:9], \text{offset}[8:0]$

Instruction Set Design

- Control:
 - Unconditional branch
 - Conditional branch
 - Procedure call and return
 - System call
- Condition bits:
 - 1-bit registers for conditional branches
 - Updated by instructions upon writing into registers
 - **N** (Negative), **P** (Positive), **Z** (Zéro).
- $JMP/JSR\ L, offset$
 - ❖ Direct addressing
 - ❖ If $L=1\ R7 \leftarrow PC$
 - ❖ $PC \leftarrow PC[15:9], offset[8:0]$
- $JMPR/JSRR\ L, Rs, base$
 - ❖ Indexed addressing
 - ❖ If $L=1\ R7 \leftarrow PC$
 - ❖ $PC \leftarrow Rs + base$
- $BR\ nzp\ offset$
 - ❖ Direct addressing
 - ❖ $PC \leftarrow PC[15:9], offset[8:0]$
 - ❖ Test condition registers if n, z and/or p bit is 1
 - ✓ Condition value = $N \& n \mid P \& p \mid Z \& z$
- RET
 - ❖ $PC \leftarrow R7$
- $TRAP\ trapvect8$
 - ❖ Call a system subroutine
 - ❖ $R7 \leftarrow PC$
 - ❖ $PC \leftarrow @(00000000trapvec8)$

Instruction Set Format



Opcode	Instruction
0000	BR
0100	JMP/JSR
1100	JMPR/JSRR
1111	TRAP
1101	RET

Contrôle

Opcode	Instruction
0010	LD
0110	LDR
1110	LEA
0011	ST
0111	STR

Mémoire

Opcode	Instruction
0001	ADD
0101	AND
1001	NOT

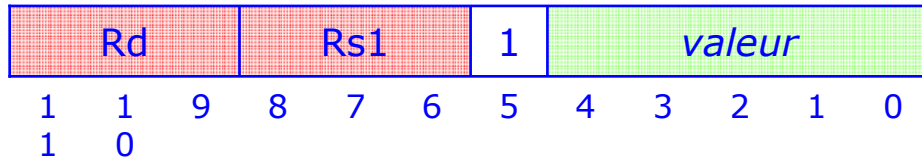
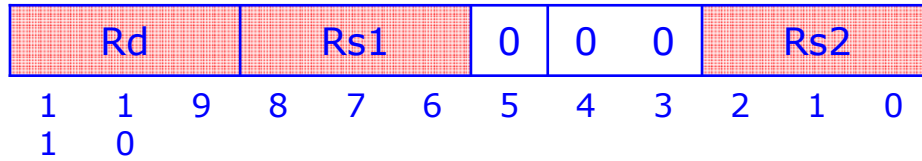
ALU

- ≤ 16 instructions \rightarrow 4 opcode bits.
- Opcode:
 - Simplify control circuit
 - Possible ISA extensions

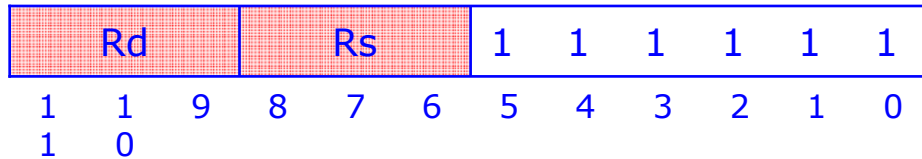
Instruction Set Format

- 8 registers → 3 bits.

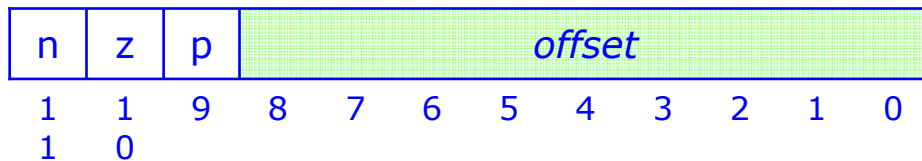
- ADD, AND:



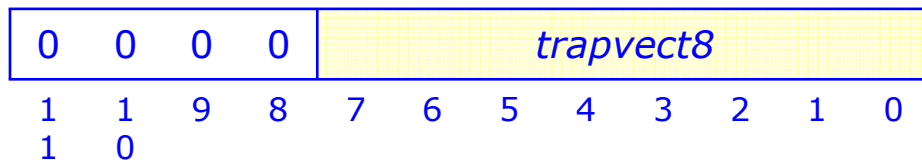
- NOT:



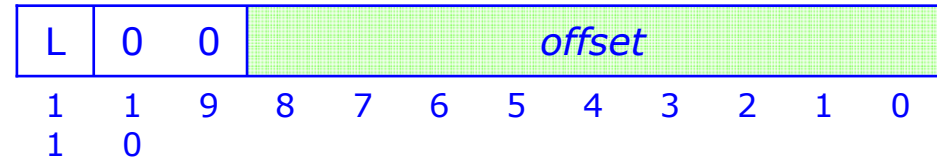
- BR:



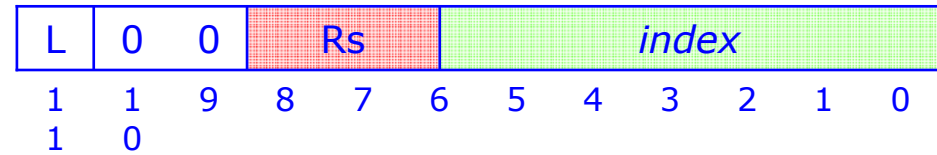
- TRAP:



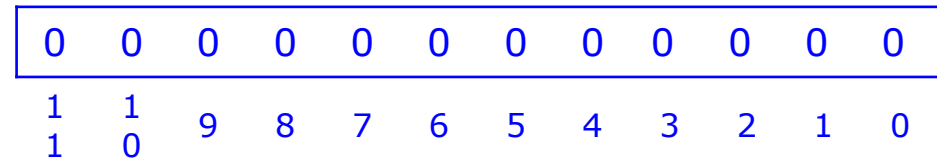
- JSR/JMP:



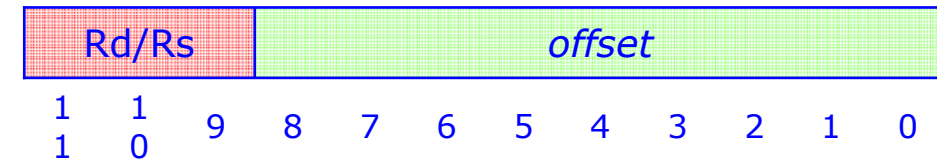
- JSRR/JMPR:



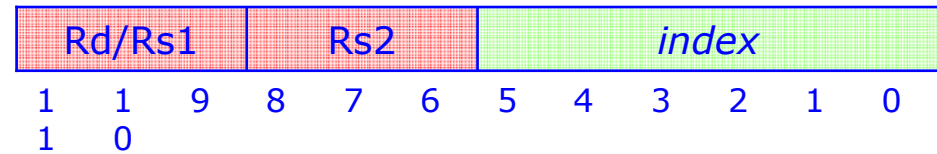
- RET:



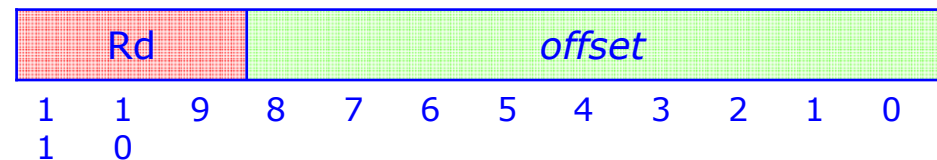
- LD/ST:



- LDR/STR:



- LEA:



Processor Architecture

- Instruction execution stages (1 stage = 1 processor cycle):
 1. Send instruction address
 2. Fetch instruction
 3. Store instruction
 4. Decode (fetch operands)
 5. Compute address
 6. Fetch operands from memory
 7. Execution
 8. Write result

ADD R5, R4, #3

LC-2

