Tema 2. Instruccions i tipus de dades bàsics

Estructura de Computadors (EC)

Rubèn Tous

rtous@ac.upc.edu
Computer Architecture Department
Universitat Politecnica de Catalunya

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índex

Introducció

- A IC disseny d'un computador simple (SISP) + relació entre el llenguatge màquina i el disseny a nivell de circuits lògics digitals.
- A EC estudiarem la relació entre el llenguatge màquina i un llenguatge d'alt nivell.
- Llenguatge màquina: MIPS de 32 bits.
- Llenguatge d'alt nivell: C.

Descripció jeràrquica del computador

Capes:

- Hardware: Dissenyador de computadors (IC).
- Llenguatge màquina (MIPS, x86, ...): Programador de sistemes (EC).
- Llenguatge d'alt nivell (C, Java, ...): Programador de software (Pro2).
- Llenguatge d'usuari (comandes, menús, ...): Usuari final.

Instruction Set Architecture (ISA)

ISA

Una Instruction Set Architecture (ISA) és una **especificació** que descriu els aspectes del processador visibles a un programador de llenguatge màquina (o assemblador): instruccions, registres, model de memòria, entrada/sortida, excepcions, etc.

- Una mateixa ISA pot ser implementada de diferents maneres.
- Compatibilitat a nivell hardware: Un programa compilat per a una ISA es podrà executar sobre qualsevol hardware que la implementi.
- Exemples: MIPS, RISC-V, x86, etc.

Application Binary Interface (ABI)

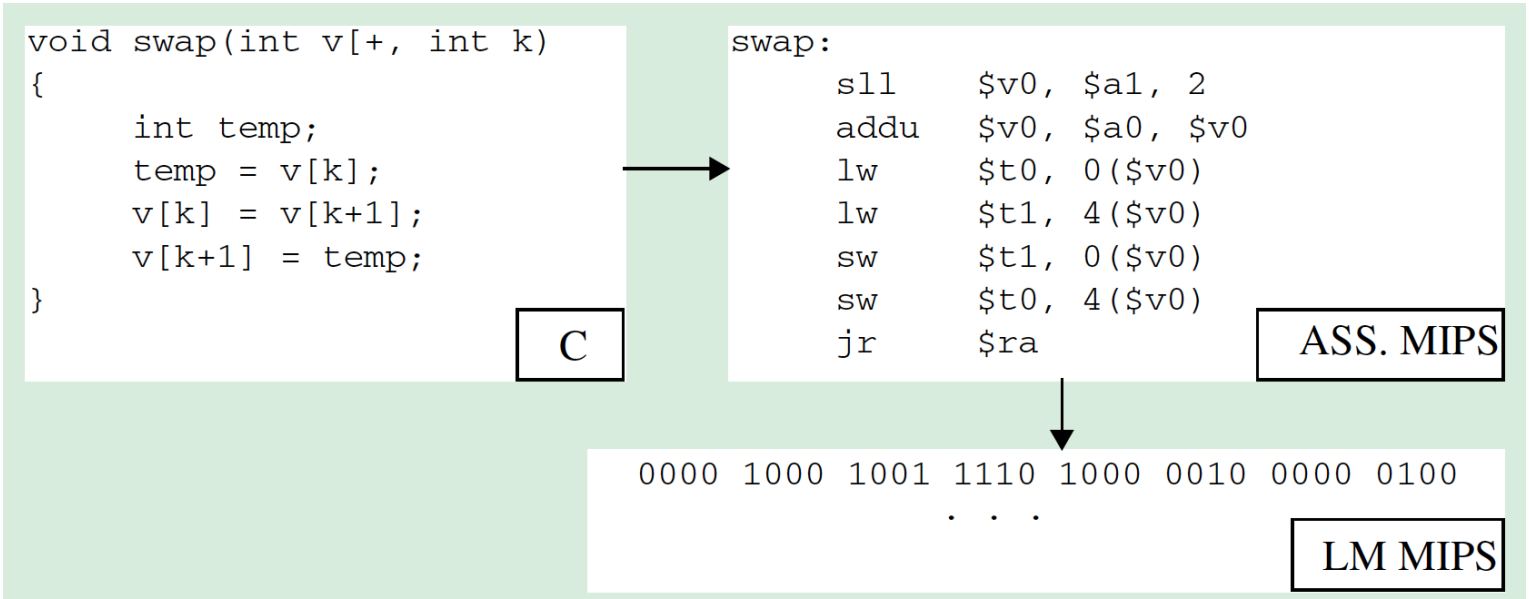
ABI

Una Application Binary Interface (ABI) és una **especificació** que descriu la interfície de baix nivell (e.g. convenis de crida i retorn de funcions) entre dos mòduls d'un programa, típicament entre un programa i les llibreries del SO.

- A EC parlarem sempre de l'ABI com a inferfície d'un SO.
- Un programa compilat per a un ISA/ABI específic podrà ser executat sense modificació sobre qualsevol plataforma que implementi aquest ISA/ABI.
- Exemples d'ABIs: System V ABI (Linux), MIPS EABI.

Traducció: Compilació vs Interpretació

- Les accions en el llenguatge de nivell i han de ser convertides al llenguatge del nivell i-1.
- El llenguatge assemblador i màquina són del mateix nivell.

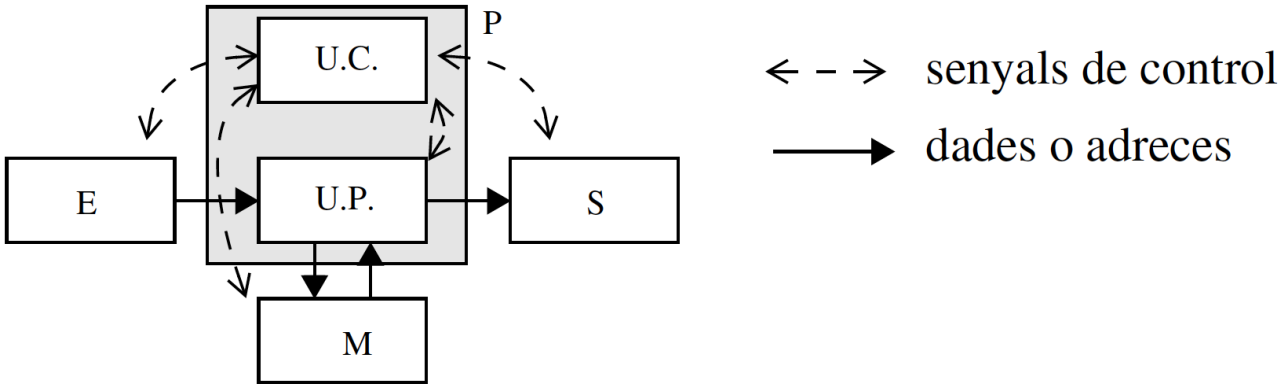


Traducció: Compilació vs Interpretació

- Compilació: Es converteix tot el programa sense executar-lo. Exemple: Compilador de C a assemblador MIPS.
  - Pros: Sols s'ha de fer 1 cop.
  - Contras: S'ha de recompilar cada cop que es vol adaptar el programa a un sistema (ISA i/o ABI) diferent.
- Interpretació: Es converteixen les accions una per una i es van executant. Exemples: Shell Unix, intèrpret Python, MV Java, Unitat de control de la CPU, etc.
  - Pros: Portabilitat, escriure un intèrpret és més senzill que escriure un compilador.
  - Contras: S'ha de traduir cada vegada, més lent.

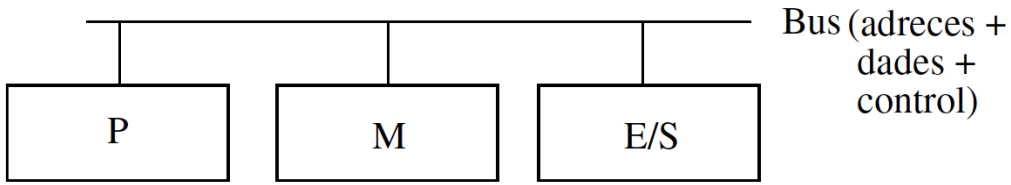
Parts d'un computador Von Neumann

- Arquitectura Von Neumann (1945).
- Instruccions del programa en el mateix espai d'adreces de memòria on es guarden les dades.



Parts d'un computador Von Neumann

- Sovint, el model es presenta de forma simplificada:



2.1 Introducció a la MIPS ISA

MIPS

MIPS és un **ISA** de tipus **RISC**.

2.1.1 Què és una arquitectura del joc d'instruccions (ISA)

ISA

Instruction Set Architecture = **Especificació** de: instruccions, registres, model de memòria, entrada/sortida, excepcions, etc.

2.1.2 Què són les arquitectures de tipus RISC?

- Complex Instruction Set Computers (CISC):
  - VLSI $\rightarrow$ permet instruccions complexes.
  - Moltes instruccions, nombre variable d'operands, mida variable.
  - Difícil optimitzar rendiment operacions simples (les més freqüents).
  - x86, PDP-11, VAX, System/360, 68000, etc.
- Reduced Instruction Set Computer (RISC):
  - Instruccions de mida fixa, pocs modes d'adreçament, accés a memòria load i store.
  - $\rightarrow$ Poques instruccions.
  - ARM, PowerPC, MIPS, Alpha, SPARC, etc.
- Actualment poques diferències entre CISC i RISC.

2.1.3 Què és l'arquitectura MIPS?

- MIPS = Microprocessor without Interlocked Pipeline Stages).
- Dr. John L. Hennessy (Stanford) el 1981.
- Primer processador: MIPS R2000 (MIPS-I de 32 bits).
- Estudiarem la versió MIPS32.

2.1.3 Què és l'arquitectura MIPS?

- Arquitectura RISC de 32 bits (1 word = 32 bits = 4 bytes)
- Instruccions de 32 bits, adreces de 32 bits, registres de 32 bits.
- Exemple: 0x44332211
- Memòria unificada per a instruccions i dades.
- Adreçament a nivell de byte (2^{32} bytes adreçables).
- 32 registres de 32 bits cadascun.
- Embedded systems (mòdems ADSL), consoles de jocs (Sony PlayStation 2 o la PlayStation Portable).

Exemple

Programa en alt nivell (llenguatge C)

```
1  int V[N] = {-1, -2, -3, -4, -5, -6, -7, -8, -9, -10};
2  void main() {
3     int suma = 0, i = 0;
4     while ( i < 10) {
5        suma = suma + V[i];
6        i++;
7     }
8  }
```

Exemple

Programa en baix nivell (ISA MIPS)

```
1   .data
2  V:.word -1, -2, -3, -4, -5, -6, -7, -8, -9, -10 # vector enters
3   .text
4   .globl main
5  main:
6     li $t0, 0                 # $t0 = 0 (variable suma)
7     li $t1, 0                 # $t1 = 0 (comptador del bucle i)
8     li $t2, 10                # $t2 = 10
9  bucle:
10    slt $t3, $t1, $t2
11    beq $t3, $zero, fibucle # 10 voltes
12    la $t3, V                 # $t3 = Adr. inicial de V
13    sll $t4, $t1, 2           # $t4 = 4*i (cada element ocupa 4 bytes)
14    addu $t3, $t3, $t4        # @V[i] = @V + 4*i
15    lw $t3, 0($t3)            # Llegim un element $t3 = V[i]
16    addu $t0, $t0, $t3        # Acumulem. suma = suma + V[i]
17    addiu $t1, $t1, 1         # i++
18    b bucle                   # salt incondicional
19 fibucle:
20    jr $ra
```

Preguntes?