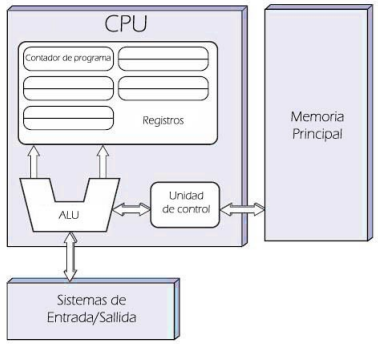
FONAMENTS D'ORDINADORS

TEMA1: Arquitectura d'un ordinador

Manel Guerrero

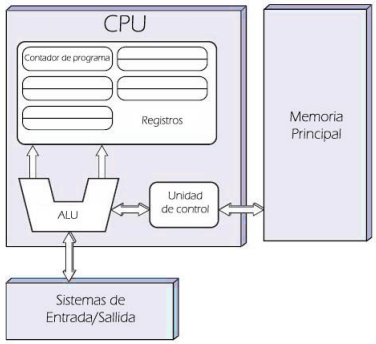
[H1] Arquitectura Von Neumann

- La majoria dels ordinadors segueixen l'estructura de Von Neumann (circa 1950): Processador, memòria i E/S connectats per un bus.
- La memòria és una matriu de bits de 'N' files i 'm' columnes.
- El bus es té tres parts:
  - dades (m bits (o 2m, o 4m, ...))
  - adreces (n bits, n=log2(N))
  - control (1 bit).



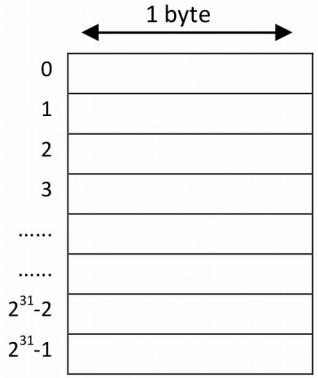
Unitat Central de Procés (CPU)

- Unitat Aritmètico Lògica (ALU): És com una mini calculadora que fa operacions aritmètiques i de lògica booleana.
- Unitat de Control (CU): interpreta les instruccions i genera els senyals de control necessaris per executar-les.
- Registres: Són com variables del processador. Llegir o escriure d'un registre és molt més ràpid que llegir o escriure a la memòria.



La memòria del IA32

- El IA32 (Intel 386 i posteriors) té la memòria organitzada en N paraules de 8 bits (1 byte). Cadascun d'aquests bytes s'adreça del 0 al N-1.
- Al ser una arquitectura de 32 bits, la mida del bus de dades/adreces i dels registres és de 32 bits.
- Quan llegeix o escriu sempre són 4 bytes en adreces múltiples de 4.
- Al tenir un bus d'adreces de 32 bits, podem adreçar 4Gigabytes (2^32 bytes).



Modelo plano de memoria

Les dades a la memòria

- Les instruccions dels programes i les dades amb les que treballen es guarden a memòria.
- Les dades:
  - Naturals (enters positius): en binari pur.
  - Caràcters: en ASCII
  - Enters amb signe: en Complement a 2 (complement a 1 + 1)

n bin	n Ca1	n Ca2
0 000	3 011	3 011
1 001	2 010	2 010
2 010	1 001	1 001
3 011	0 000	0 000
4 100	-0 111	-1 111
5 101	-1 110	-2 110
6 110	-2 101	-3 101
7 111	-3 100	-4 100

Taula ASCII

000	(nul)	016 ►	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001 ☺	(soh)	017 ◄	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002 ☻	(stx)	018 ↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003 ♥	(etx)	019 ‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004 ♦	(eot)	020 ¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005 ♣	(enq)	021 §	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006 ♠	(ack)	022 ▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007 •	(bel)	023 ↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008 ◘	(bs)	024 ↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009	(tab)	025 ↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011 ♂	(vt)	027 ←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012 ♀	(np)	028 ∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013	(cr)	029 ↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014 ♫	(so)	030 ▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015 ☼	(si)	031 ▼	(us)	047	/	063	?	079	O	095	_	111	o	127	⌂

Espai 32 | 0 48 | A 65 | a 97

Complement a 2 (*)

- Els positius es guarden en binari pur i els negatius en Ca2.
- Ca2 = Ca1 + 1
- Conversió ràpida per humans: #1: començant per la dreta copiar el número original fins el primer 1 (primer 1 inclòs). #2: Després, es neguen (complementen) els dígits restants (es a dir, substituïm 0 per 1 i viceversa).
  - `Ex: 92 -> -92`
  - `#1 01011100 -> ?????100`
  - `#2 01011100 -> 10100100`
- El Complement a 2 és molt pràctic per als ordinadors per que el bit de més pes determina el signe i podem sumar números sense pensar si són amb signe o sense i funciona.
- A més per restar A-B podem fer A+Ca2(B).
- Exemple de suma:

```
(181) 1 0 1 1 0 1 0 1 (-75)
( 59) 0 0 1 1 1 0 1 1 (+59)
+---  --------------- +---
(240) 1 1 1 1 0 0 0 0 (-16)
```

[H2] Els programes a la memòria

- Els programes són seqüències d'instruccions.
- Les instruccions CM dels programes es codifiquen com una seqüència de bytes a memòria. Cada instrucció té:
  - Codi d'Operació.
  - Operador Font #1
  - [Operador Font #2]
  - Operador Destí
- Alguns operadors poden ser implícits. Exemple:
  - `C: a = a + 2;`
  - `ASM: ADDL $2, %EAX`
  - `CM: 0x 83 C0 02`
    - `CO: 83 (ADDL)`
    - `OF1: 02`
    - `OF2: C0 (%EAX)`
    - `OD: C0.`

Execució d'un programa

- El S.O. carrega el programa a memòria.
- El registre %eip guarda el valor de la primera instrucció a executar.
- La CPU va executant les instruccions en ordre seqüencial
- %eip sempre guarda el valor de la següent instrucció a executar.
- Existeixen instruccions que poden trencar l'execució seqüencial: salts i crides a subrutines).
- El programa acaba amb una instrucció especial que retornen el control al S.O.

- Per cada instrucció la CPU fa:
  - Fetch: Anar a buscar a memòria la instrucció a executar.
  - Descodificació: Determinar el tipus d'operació a realitzar i on es troben els operands.
  - Obtenir operands font.
  - Execució.
  - Escriure el resultat a l'operand destí.
  - Incremento %eip.
  - En anglès: "fetch, decode, execute, and store".

Algorismes i programes

- Algorisme: seqüència d'instruccions, no ambigües, l'execució de les quals condueix a una resolució d'un problema.
- L'algorisme pot rebre una "entrada"
- I donar com a resultat una "sortida" (la solució del problema).
- Programa: Algorisme codificat (escrit) en un llenguatge de programació determinat.

Estructura d'un computador

- El processador central executa programes
  - que llegeixen de dispositius d'entrada
    - (com ara el teclat)
  - i escriuen a dispositius de sortida
    - (com ara la pantalla)
  - pot emmagatzemar dades en memòria RAM
    - Memòria volàtil (memòria RAM)
  - I llegeixen/escriuen dades en dispositius E/S
    - (Com ara un disc dur)

Llenguatges de programació

- D'alt nivell (propers a com pensem els humans). Com ara el "C". Per qualsevol ordinador. `[ a = a + 2; ]`
- De baix nivell (propers a com funcionen un tipus de computadora en concret) com ara el llenguatge assemblador del I32. `[ ADDL $2, %EAX ]`
- Codi màquina: Seqüència de '0's i '1's en que un programa es codifica perquè un ordinador en concret ho pugui executar. `[ Hex: 83 C0 02 ]`

Elaboració d'un programa: Etapes

- Anàlisi: Definir bé el problema (ex: programa que donada una base i una alçada calculi l'àrea d'un triangle rectangle).
- Disseny: Desenvolupar l'algorisme
- Codificació: Escriure el codi que l'implementi
- Proba: Execució del codi amb diferents entrades.

- Manteniment: Millora del codi, esmena de nous errors que es van detectant.