

Representación da Información

Sabemos que os sistemas de memoria almacenan só números binarios. Pero os usuarios non só queremos almacenar números. Queremos almacenar texto, imaxes, vídeo....

Texto

Para almacenar texto precisamos un código no que un conxunto de ceros e uns preestablecidos se correspondan a un símbolo. **Ese código é o ASCII.**

Caracteres de control ASCII				Caracteres ASCII imprimibles									ASCII extendido														
DEC	HEX	Símbolo ASCII		DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
00	00h	NULL (carácter nulo)		32	20h	espacio	64	40h	@	96	60h	`	128	80h	Ç	160	A0h	á	192	C0h	Ł	224	E0h	Ó			
01	01h	SOH (inicio encabezado)		33	21h	!	65	41h	A	97	61h	a	129	81h	ü	161	A1h	î	193	C1h	ł	225	E1h	ô			
02	02h	STX (inicio texto)		34	22h	"	66	42h	B	98	62h	b	130	82h	é	162	A2h	ó	194	C2h	Ł	226	E2h	ö			
03	03h	ETX (fin de texto)		35	23h	#	67	43h	C	99	63h	c	131	83h	â	163	A3h	û	195	C3h	ł	227	E3h	ø			
04	04h	EOT (fin transmisión)		36	24h	\$	68	44h	D	100	64h	d	132	84h	ä	164	A4h	ü	196	C4h	Ł	228	E4h	õ			
05	05h	ENQ (enquiry)		37	25h	%	69	45h	E	101	65h	e	133	85h	å	165	A5h	ñ	197	C5h	ł	229	E5h	ö			
06	06h	ACK (acknowledgement)		38	26h	&	70	46h	F	102	66h	f	134	86h	ä	166	A6h	ª	198	C6h	Ł	230	E6h	µ			
07	07h	BEL (timbre)		39	27h	'	71	47h	G	103	67h	g	135	87h	ç	167	A7h	º	199	C7h	ł	231	E7h	þ			
08	08h	BS (retroceso)		40	28h	(72	48h	H	104	68h	h	136	88h	ê	168	A8h	ë	200	C8h	Ł	232	E8h	ÿ			
09	09h	HT (tab horizontal)		41	29h)	73	49h	I	105	69h	i	137	89h	ë	169	A9h	¸	201	C9h	ł	233	E9h	ÿ			
10	0Ah	LF (salto de línea)		42	2Ah	*	74	4Ah	J	106	6Ah	j	138	8Ah	è	170	AAh	¸	202	CAh	Ł	234	EAh	ÿ			
11	0Bh	VT (tab vertical)		43	2Bh	+	75	4Bh	K	107	6Bh	k	139	8Bh	ï	171	ABh	½	203	CBh	ł	235	EBh	ÿ			
12	0Ch	FF (form feed)		44	2Ch	,	76	4Ch	L	108	6Ch	l	140	8Ch	î	172	ACH	¼	204	CCh	Ł	236	ECh	ÿ			
13	0Dh	CR (retorno de carro)		45	2Dh	-	77	4Dh	M	109	6Dh	m	141	8Dh	ï	173	ADh	»	205	CDh	ł	237	EDh	ÿ			
14	0Eh	SO (shift Out)		46	2Eh	.	78	4Eh	N	110	6Eh	n	142	8Eh	Ā	174	A Eh	«	206	CEh	Ł	238	EEh	ÿ			
15	0Fh	SI (shift in)		47	2Fh	/	79	4Fh	O	111	6Fh	o	143	8Fh	Ā	175	AFh	»	207	CFh	ł	239	EFh	ÿ			
16	10h	DLE (data link escape)		48	30h	0	80	50h	P	112	70h	p	144	90h	Ē	176	B0h	⋮	208	D0h	Ł	240	F0h	±			
17	11h	DC1 (device control 1)		49	31h	1	81	51h	Q	113	71h	q	145	91h	æ	177	B1h	⋮	209	D1h	ł	241	F1h	±			
18	12h	DC2 (device control 2)		50	32h	2	82	52h	R	114	72h	r	146	92h	Æ	178	B2h	⋮	210	D2h	Ł	242	F2h	¼			
19	13h	DC3 (device control 3)		51	33h	3	83	53h	S	115	73h	s	147	93h	ø	179	B3h	⋮	211	D3h	ł	243	F3h	¼			
20	14h	DC4 (device control 4)		52	34h	4	84	54h	T	116	74h	t	148	94h	ò	180	B4h	⋮	212	D4h	Ł	244	F4h	¾			
21	15h	NAK (negative acknowle.)		53	35h	5	85	55h	U	117	75h	u	149	95h	ó	181	B5h	⋮	213	D5h	ł	245	F5h	¾			
22	16h	SYN (synchronous idle)		54	36h	6	86	56h	V	118	76h	v	150	96h	ü	182	B6h	⋮	214	D6h	Ł	246	F6h	÷			
23	17h	ETB (end of trans. block)		55	37h	7	87	57h	W	119	77h	w	151	97h	ÿ	183	B7h	⋮	215	D7h	ł	247	F7h	÷			
24	18h	CAN (cancel)		56	38h	8	88	58h	X	120	78h	x	152	98h	ÿ	184	B8h	⋮	216	D8h	Ł	248	F8h	÷			
25	19h	EM (end of medium)		57	39h	9	89	59h	Y	121	79h	y	153	99h	ÿ	185	B9h	⋮	217	D9h	ł	249	F9h	÷			
26	1Ah	SUB (substitute)		58	3Ah	:	90	5Ah	Z	122	7Ah	z	154	9Ah	ÿ	186	BAh	⋮	218	DAh	Ł	250	FAh	÷			
27	1Bh	ESC (escape)		59	3Bh	;	91	5Bh	[123	7Bh	{	155	9Bh	ø	187	BBh	⋮	219	DBh	ł	251	FBh	÷			
28	1Ch	FS (file separator)		60	3Ch	<	92	5Ch	\	124	7Ch		156	9Ch	£	188	BCh	⋮	220	DCh	Ł	252	FCh	÷			
29	1Dh	GS (group separator)		61	3Dh	=	93	5Dh]	125	7Dh	}	157	9Dh	Ø	189	BDh	⋮	221	DDh	ł	253	FDh	÷			
30	1Eh	RS (record separator)		62	3Eh	>	94	5Eh	^	126	7Eh	~	158	9Eh	x	190	BEh	⋮	222	DEh	Ł	254	FEh	÷			
31	1Fh	US (unit separator)		63	3Fh	?	95	5Fh	_				159	9Fh	f	191	BFh	⋮	223	DFh	ł	255	FFh	÷			
127	20h	DEL (delete)								elCodigoASCII.com.ar																	

Cada carácter ocupa 8 bits en código ASCII polo que temos 256 posibles símbolos.

Exemplo:

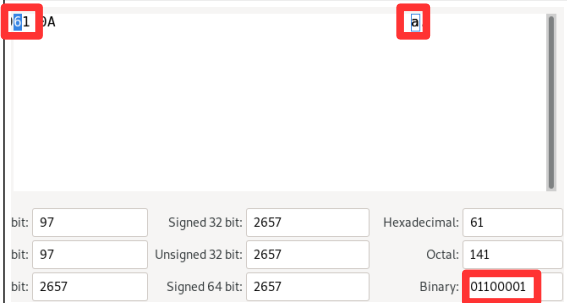
En Windows executa un editor de texto ASCII como notepad. Sen ter activo o teclado numérico escribe

- ALT Esquerdo + 65
- ALT Esquerdo + 97
- ALT Esquerdo + 126

Cal é a razón pola que se amosan eses caracteres?

Exemplo:

Creamos un arquivo con unha letra a, Se o examinamos cun editor hexadecimal:



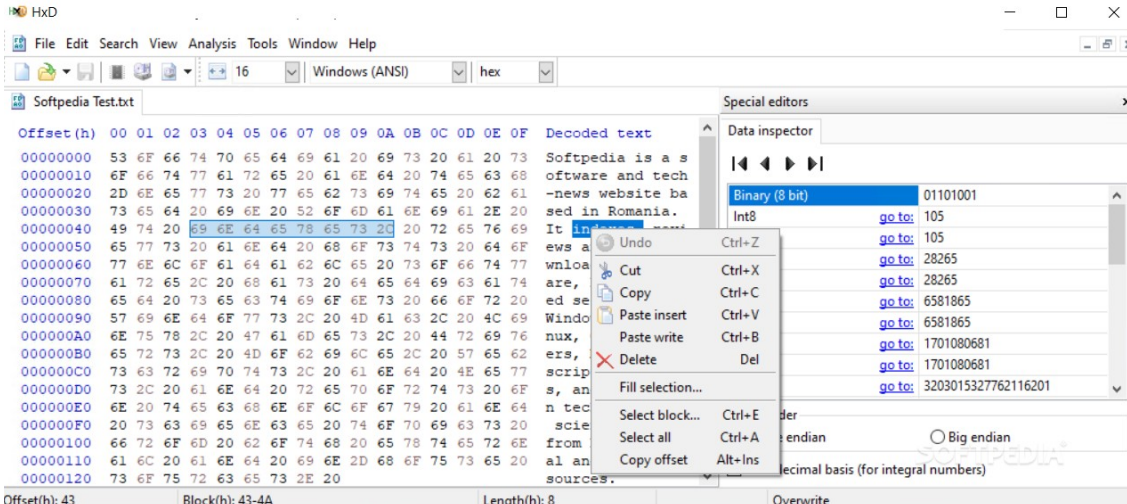
Amosa o contido en:

- Hexadecimal: 61
- Código ASCII: a
- Binario: 01100001

Exemplo:

Descarga a versión portable do programa [HxD20](#).

- Escribe un documento de texto que conteña o seguinte texto.
“Este camión é moi grande”
- Edita o arquivo que acabas de crear co editor hexadecimal e modifícao para que en “Este” lugar dunha e maiúscula apareza unha e minúscula.



Offset(h): 43 Block(h): 43-4A Length(h): 8 Overwrite



Exercicio: Empregando un editor hexadecimal en GNU/Linux

- Inicia sesión en GNU/Linux
- Crea un arquivo co gedit que conteña o carácter a
- Representación de a en varias codificacións
a = ASCII 97(d 61(h 11000001(2
- Examina o seu contido co editor hexadecimal en liña de comandos **xxd**
 - Observamos o valor do arquivo en Binario

```
rojas@debian:~/Imaxes/Temp$ xxd -b f1.txt
```

```
00000000: 01100001 00001010                                     a.
```

- Observamos o valor do arquivo en Hexadecimal

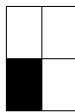
```
rojas@debian:~/Imaxes/Temp$ xxd f1.txt
```

```
00000000: 610a                                                  a.
```

Imaxes

Supoñamos que queremos almacenar unha imaxe en formato .bmp. Teremos que almacenar cada un dos pixeles que a forman.

Supoñamos que queremos almacenar a seguinte imaxe



- As súas dimensións son 2x2 píxeles
- Como é en branco e negro só necesitamos almacenar 1 bit por pixel

O formato do arquivo .bmp podería ser o seguinte

0002 _(h)	0002 _(h)	0001 _(h)	1	1	0	1
Res. Horiz	Res. Vert	Bits/pixel				



Exercicio: .

- Os campos resolución Horizontal e Vertical son de 16 bits. Cales son as dimensións máximas que podería alcanzar unha imaxe almacenada así?

COMPILAR UN PROGRAMA

- Creamos o seguinte programa e **linguaxe C**, chamado programa.c

```
#include <stdio.h>
int main(void)
{
    int a=1;
    int b=2;
    int c=a+b;
    printf("A suma é %d\n",c);
    return 0;
}
```

- Compilamos o programa e xeramos o executable.

```
gcc pru.c --> ./a.out
```

- Examinamos o contido do executable en formato Binario.

```
rojas@debian:~$ xxd -b a.out
```

```
0000000: 01111111 01000101 01001100 01000110 00000010 00000001 .ELF..
0000006: 00000001 00000000 00000000 00000000 00000000 00000000 .....
000000c: 00000000 00000000 00000000 00000000 00000010 00000000 .....
0000012: 00111110 00000000 00000001 00000000 00000000 00000000 >.....
0000018: 00000000 00000100 01000000 00000000 00000000 00000000 ..@...
000001e: 00000000 00000000 01000000 00000000 00000000 00000000 ..@...
0000024: 00000000 00000000 00000000 00000000 01001000 00001010 ....H.
000002a: 00000000 00000000 00000000 00000000 00000000 00000000 .....
0000030: 00000000 00000000 00000000 00000000 01000000 00000000 ....@.
0000036: 00111000 00000000 00001000 00000000 01000000 00000000 8...@.
000003c: 00011101 00000000 00011100 00000000 00000110 00000000 .....
0000042: 00000000 00000000 00000101 00000000 00000000 00000000 .....
0000048: 01000000 00000000 00000000 00000000 00000000 00000000 @.....
000004e: 00000000 00000000 01000000 00000000 01000000 00000000 ..@.@.
0000054: 00000000 00000000 00000000 00000000 01000000 00000000 ....@.
000005a: 01000000 00000000 00000000 00000000 00000000 00000000 @.....
0000060: 11000000 00000001 00000000 00000000 00000000 00000000 .....
0000066: 00000000 00000000 11000000 00000001 00000000 00000000 .....
000006c: 00000000 00000000 00000000 00000000 00001000 00000000 .....
```

- Examinamos o contido do executable en linguaxe ensamblador.

gcc -S pru.c --> pru.s **Código en ensamblador**

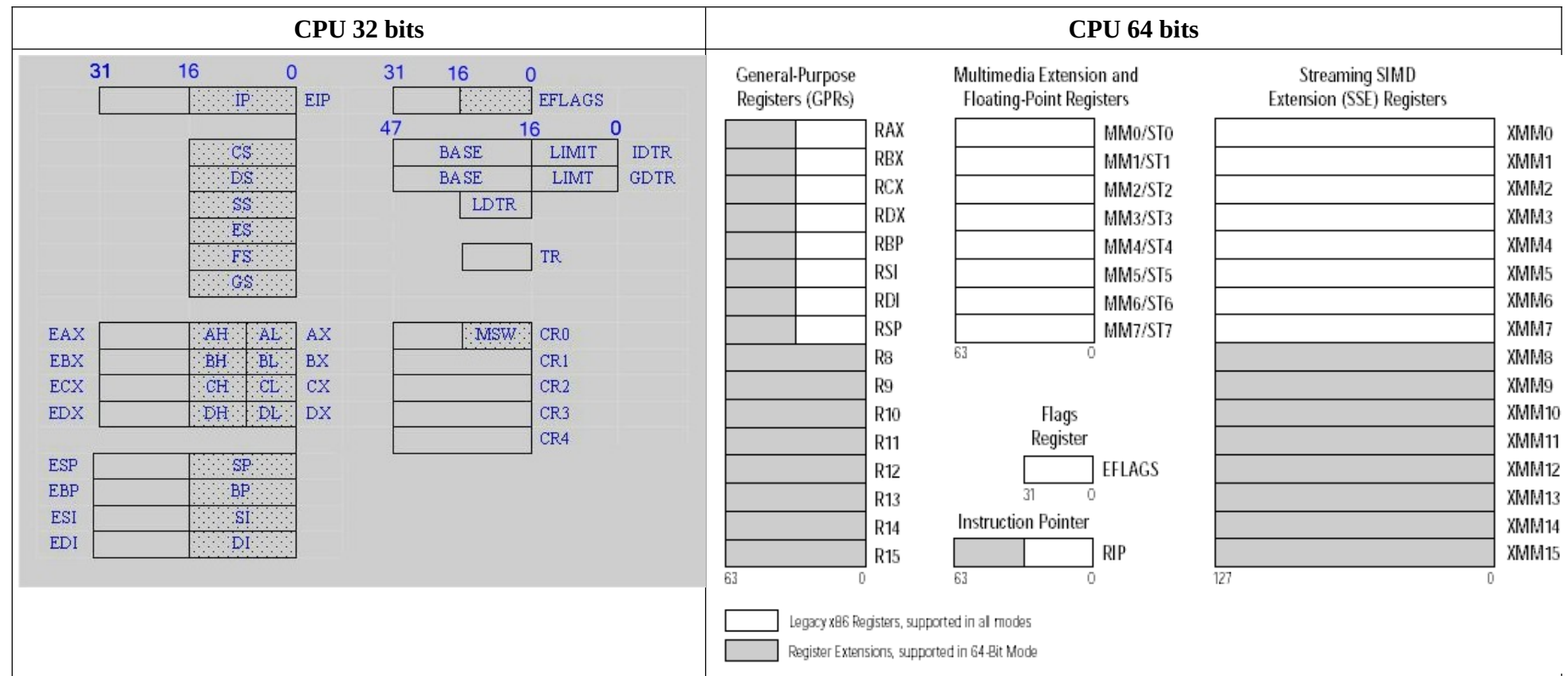
```
.file "pru.c"
.text
.globl      main
.type main, @function
main:
.LFB0:
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
movl $1, -4(%rbp)
movl $2, -8(%rbp)
movl -8(%rbp), %eax
movl -4(%rbp), %edx
addl %edx, %eax
movl %eax, -12(%rbp)
movl $0, %eax
popq %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size main, .-main
.ident      "GCC: (Debian 4.7.2-5) 4.7.2"
.section    .note.GNU-stack,"",@progbits
```



Exercicio: .

- Que son eax e edx? Emprega a imaxe da páxina seguinte para telo máis claro
- Que é o que realmente fai o programa para sumar dous números?

Os registros da CPU



As arquitecturas son retrocompatibles. Unha CPU de 64 bits pode executar programas de 32, 16 ...