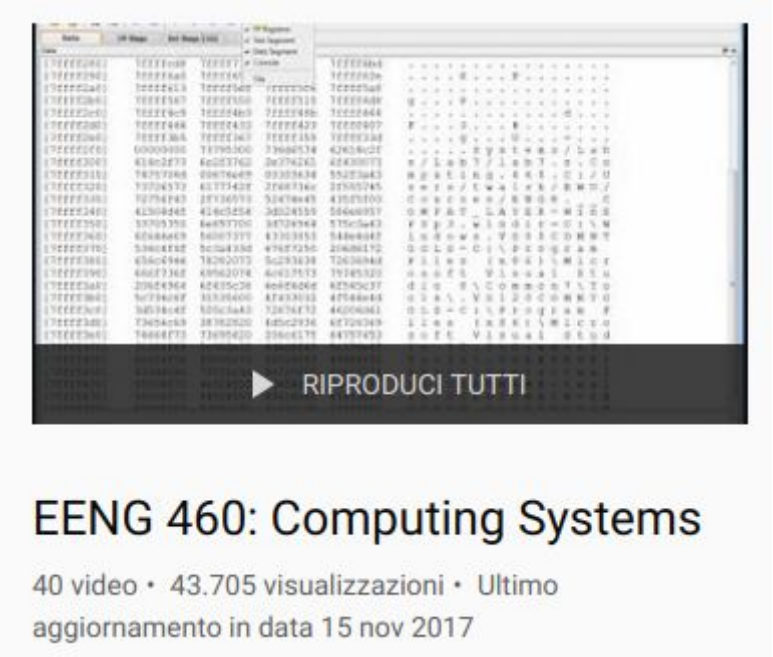
15 Maggio 2020

Risorse Utili

Tutorato Architettura degli Elaboratori, Lezione 4

- Il link per scaricare ed installare QtSpim è il seguente: [http://spimsimulator.sourceforge.net/] è disponibile sia una versione per Windows, sia per distribuzioni Linux che per Mac.

- Se siete in difficoltà e non sapete programmare bene in assembly o usare QtSpim, vi consiglio **caldamente** questa playlist dove un Prof della Eastern Washington University (EWU) insegna le basi dell'assembly MIPS.
  [https://www.youtube.com/playlist?list=PLW7Cvy3HywwwiTivCN8jm2yDn9N5Eyxza]



  p.s.
  consiglio di guardare la playlist anche se non siete in difficoltà.

- Come text editor vi consiglio Visual Studio Code [https://code.visualstudio.com/] ora è "l'editor del momento" in quanto leggero, pieno di plug-in, veloce ed esteticamente appealing. C'è sia per Windows che per Linux che per Mac. Quello che mi avete visto usare a tutorato è NeoVim [https://neovim.io/].

Di seguito inserisco le figure che ho utilizzato durante la lezione.

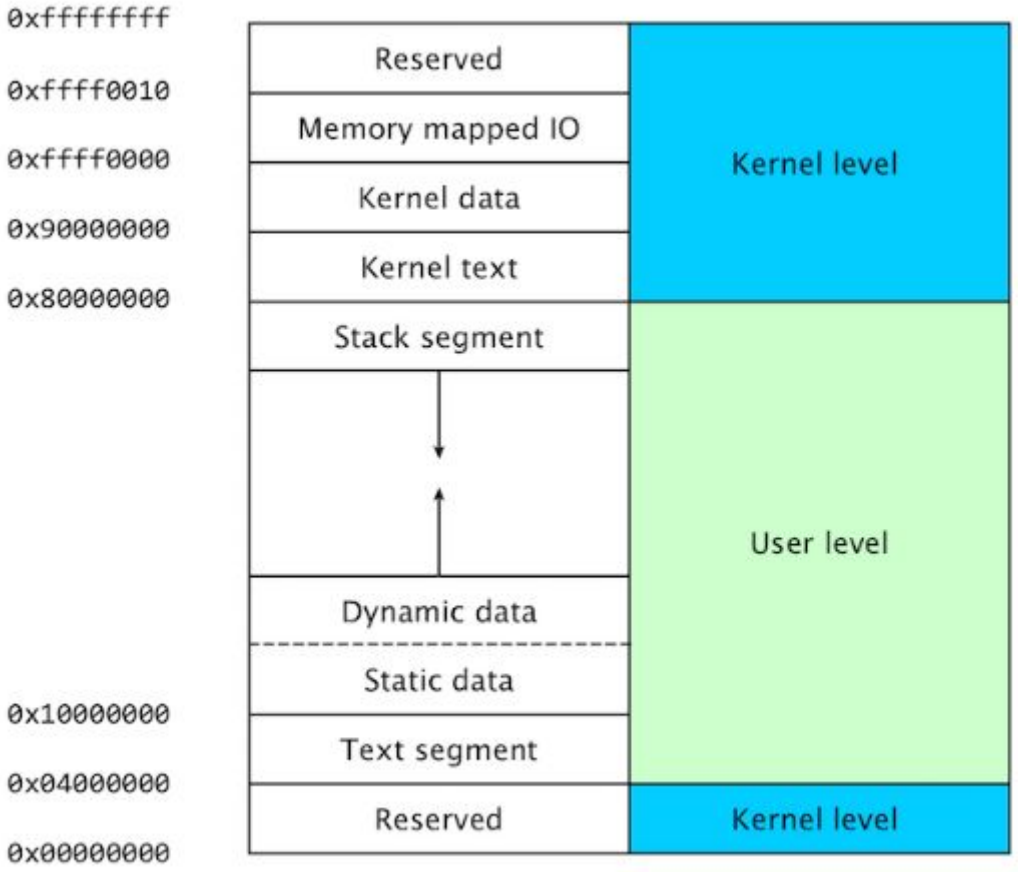
Tabella ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Source: http://www.asciitable.com/

MIPS Memory Layout



Source:
http://www.it.uu.se/education/course/homepage/os/vt18/module-0/mips-and-mars/mips-memory-layout/

Convenzioni Registri MIPS

Nome del registro	Numero	Utilizzo
$zero	0	costante 0
$at	1	riservato all'assemblatore
$v0	2	calcolo di espressioni e risultati di una funzione
$v1	3	calcolo di espressioni e risultati di una funzione
$a0	4	argomento 1
$a1	5	argomento 2
$a2	6	argomento 3
$a3	7	argomento 4
$t0	8	temporaneo (non preservato dalle procedure)
$t1	9	temporaneo (non preservato dalle procedure)
$t2	10	temporaneo (non preservato dalle procedure)
$t3	11	temporaneo (non preservato dalle procedure)
$t4	12	temporaneo (non preservato dalle procedure)
$t5	13	temporaneo (non preservato dalle procedure)
$t6	14	temporaneo (non preservato dalle procedure)
$t7	15	temporaneo (non preservato dalle procedure)
$s0	16	temporaneo salvato (preservato dalle procedure)
$s1	17	temporaneo salvato (preservato dalle procedure)
$s2	18	temporaneo salvato (preservato dalle procedure)
$s3	19	temporaneo salvato (preservato dalle procedure)
$s4	20	temporaneo salvato (preservato dalle procedure)
$s5	21	temporaneo salvato (preservato dalle procedure)
$s6	22	temporaneo salvato (preservato dalle procedure)
$s7	23	temporaneo salvato (preservato dalle procedure)
$t8	24	temporaneo (non preservato dalle procedure)
$t9	25	temporaneo (non preservato dalle procedure)
$k0	26	riservato per il kernel del sistema operativo
$k1	27	riservato per il kernel del sistema operativo
$gp	28	puntatore all'area globale
$sp	29	stack pointer (puntatore allo stack)
$fp	30	frame pointer (puntatore al frame della procedura)
$ra	31	indirizzo di ritorno (utilizzato dalla chiamata a procedura)

Figura A.6.1. I registri MIPS e le loro convenzioni di utilizzo.

Source: Struttura e progetto dei calcolatori - David A. Paterson, John L. Hennessy, Appendice A

Codici Syscall

Funzione di Servizio	Codice di chiamata di sistema	Argomenti	Risultato
print_int	1	$a0 = intero	
print_float	2	$f12 = virgola mobile, singola pr.	
print_double	3	$f12 = virgola mobile, doppia pr.	
print_string	4	$a0 = stringa	
read_int	5		Intero (in $v0)
read_float	6		Virgola mobile, singola pr. (in $v0)
read_double	7		Virgola mobile, doppia pr. (in $v0)
read_string	8	$a0 = buffer, $a1 = lunghezza	
sbrk	9	$a0 = quantità	Indirizzo (in $v0)
exit	10		
print_char	11	$a0 = carattere	
read_char	12		Carattere (in $v0)
open	13	$a0 = nome file (stringa), $a1 = flags, $a2 = modalità	Descrittore del file (in $a0)
read	14	$a0 = descrittore del file, $a1 = buffer, $a2 = lunghezza	Num. caratteri letti (in $a0)
write	15	$a0 = descrittore file, $a1 = buffer, $a2 = lunghezza	Num. caratteri scritti (in $a0)
close	16	$a0 = descrittore del file	
exit2	17	$a0 = risultato	

Figura A.9.1. Funzioni di servizio di sistema.

Source: Struttura e progetto dei calcolatori - David A. Paterson, John L. Hennessy, Appendice A

Tipi dichiarabili nel .data segment

SPIM supporta un sottoinsieme delle direttive MIPS per l'assemblatore:

`.align n`	Allinea il prossimo dato a un confine di 2^n byte. Per esempio, `.align 2` allinea il valore successivo al confine di una parola. `.align 0` disattiva l'allineamento automatico delle direttive `.half`, `.word`, `.float` e `.double`, fino alla direttiva `.data` o `.kdata` successiva.
`.ascii str`	Salva la stringa `str` in memoria ma non la termina con il carattere null.
`.asciiz str`	Memorizza la stringa `str` in memoria e la termina con il carattere null.
`.byte b1, ..., bn`	Memorizza gli *n* valori in byte successivi della memoria.
`.data <addr>`	Gli elementi seguenti vengono memorizzati nel segmento dati. Se è presente l'argomento opzionale `addr`, gli elementi che seguono vengono memorizzati a partire dall'indirizzo `addr`.
`.double d1, ..., dn`	Memorizza gli *n* numeri in virgola mobile in doppia precisione in locazioni di memoria consecutive.
`.extern sym size`	Dichiara che il dato memorizzato in corrispondenza di `sym` occupa `size` byte ed è un'etichetta globale. Questa direttiva permette all'assemblatore di memorizzare il dato in una porzione del segmento dati cui si può accedere in modo efficiente per mezzo del registro `$gp`.
`.float f1, ..., fn`	Memorizza gli *n* numeri in virgola mobile in singola precisione in locazioni di memoria consecutive.
`.globl sym`	Dichiara che l'etichetta `sym` è globale e che si può fare riferimento a essa da altri file.
`.half h1, ..., hn`	Memorizza gli *n* numeri a 16 bit in locazioni di memoria consecutive.
`.kdata <addr>`	I dati che seguono vengono memorizzati nel segmento dati del kernel. Se è presente l'argomento opzionale `addr`, gli elementi che seguono vengono memorizzati a partire dall'indirizzo `addr`.
`.ktext <addr>`	Gli elementi che seguono vengono memorizzati nel segmento testo del kernel. In SPIM, tali elementi possono essere soltanto istruzioni o parole (si veda la direttiva `.word`). Se è presente l'argomento opzionale `addr`, gli elementi che seguono vengono memorizzati a partire dall'indirizzo `addr`.
`.set noat` e `.set at`	La prima direttiva evita che SPIM segnali che le istruzioni successive possano utilizzare il registro `$at`.

	La seconda direttiva riabilita questa segnalazione. Dato che le pseudoistruzioni vengono espanse in codice che utilizza il registro `$at`, i programmatori devono essere molto cauti nel lasciare dei valori in tale registro.
`.space n`	Alloca *n* byte di spazio nel segmento corrente (che, in SPIM, deve essere necessariamente il segmento dati).
`.text <addr>`	Gli elementi che seguono vengono posti nel segmento testo. In SPIM, tali elementi possono essere soltanto istruzioni o parole (si veda la direttiva `.word`). Se è presente l'argomento opzionale `addr`, gli elementi che seguono vengono memorizzati a partire dall'indirizzo `addr`.
`.word w1, ..., wn`	Memorizza le *n* variabili a 32 bit in parole di memoria successive.

SPIM non fa distinzione fra le diverse parti del segmento dati (`.data`, `.rdata` e `.sdata`).

Source: Struttura e progetto dei calcolatori - David A. Paterson, John L. Hennessy, Appendice A