Università degli Studi di Milano
Corso di Laurea in
Informatica per la Comunicazione Digitale
Architettura degli Elaboratori

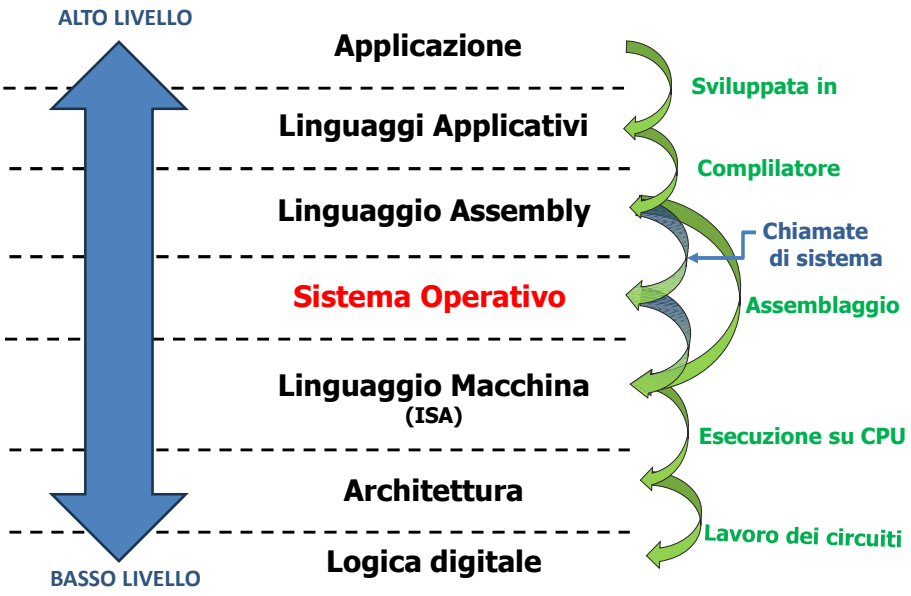
Esempi di programmi in MIPS

Chiamate di sistema e Sistema Operativo

Marco Tarini

124

Livelli di astrazione: un livello intermedio



126

Il Sistema Operativo in breve

- Software di base che gira sempre in background (finché l'elaboratore è acceso)
- Avvia ed esegue programmi su richiesta dell'utente
- Gestisce le risorse hardware (RAM, disco, periferiche)
- Alterna l'esecuzione di programmi diversi sulla CPU («multitasking»)
- Comprende i drivers: software fornito dai produttori per usare le specifiche periferiche
- Comprende le funzioni per gestire eccezioni e interrupt

127

Servizi e Componenti del Sistema Operativo

Gestione delle eccezioni e degli interrupt

Offre **servizi** ai programmi tramite **system call**, per

- Accesso al file system (lettura/scrittura sulla memoria di massa)
- Gestione della memoria centrale (RAM) (allocazione e deallocazione dinamica)
- Gestione delle periferiche, comprese di I/O: schermo, tastiera, rete, mouse ecc.
- Terminazione del programma

I Sistemi Operativi verranno studiati in dettaglio nell'omonimo corso! (del II° semestre)

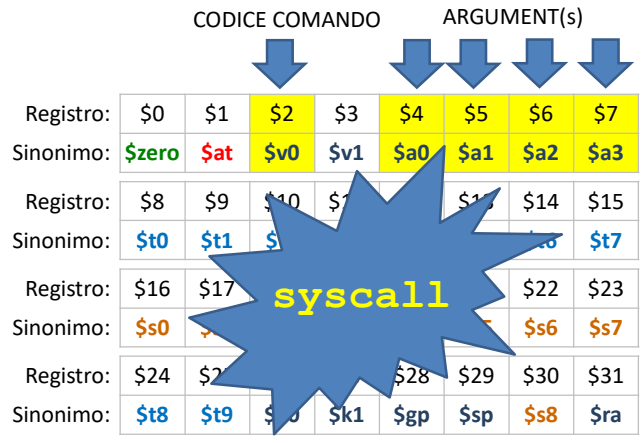
128

System Calls in MIPS-32

- **System call**: permette di utilizzare **servizi** la cui esecuzione è a carico del sistema operativo: come le operazioni di input/output e di interfacciamento con le periferiche (attraverso i drivers)
  - Nota: anche la terminazione del programma è una system call
  - Niente distingue l'ultima riga del nostro codice dalla parola memorizzata nella locazione di RAM successiva: la CPU non ha modo (da sola) di capire che il programma è terminato
- Ogni servizio è associato ad un codice numerico univoco (un intero **K**)
  - Su MIPSWEB: la lista è disponibile da help -> system calls
- Come si utilizza una system call che ha una dato **CODICE** (numerico)?
  1. caricare il **CODICE** nell'apposito registro `$v0`
  2. caricare gli argomenti (se necessari) negli appositi registri `$a0, $a1, $a2, $a3`
  3. eseguire l'istruzione `syscall`
  4. leggere eventuali valori di ritorno nei registri `$v0` (e, `$v1`).

129

Chiamata della System Call



131

Dopo la chiamata della System Call

OUTPUT(s)

Registro:	$0	$1	$2	$3	$4	$5	$6	$7
Sinonimo:	$zero	$at	$v0	$v1	$a0	$a1	$a2	$a3
Registro:	$8	$9	$10	$11	$12	$13	$14	$15
Sinonimo:	$t0	$t1	$t2	$t3	$t4	$t5	$t6	$t7
Registro:	$16	$17	$18	$19	$20	$21	$22	$23
Sinonimo:	$s0	$s1	$s2	$s3	$s4	$s5	$s6	$s7
Registro:	$24	$25	$26	$27	$28	$29	$30	$31
Sinonimo:	$t8	$t9	$k0	$k1	$fp	$sp	$s8	$ra

132

System Calls «Canoniche»

Syscall	Codice	Argomenti	Valore di ritorno	Descrizione
print_int	1	intero da stampare in $a0	nessuno	Stampa l'intero passato in $a0
print_float	2	float da stampare in $f12	nessuno	Stampa il float passato in $f12
print_double	3	double da stampare in $f12	nessuno	Stampa il double passato in $f12
print_string	4	Indirizzo della stringa da stampare in $a0	nessuno	Stampa la stringa che sta all'indirizzo passato in $a0
read_int	5	nessuno	Intero letto in $v0	Legge un intero in input e lo scrive in $v0
read_float	6	nesuno	Float letto in $f0	Legge un float in input e lo scrive in $f0
read_double	7	nessuno	Double letto in $f0	Legge un double in input e lo scrive in $f0
read_string	8	Indirizzo nel segmento dati a cui salvare la stringa in $a0 e lunghezza in byte in $a1	nessuno	Legge una stringa di lunghezza specificata in $a1 e la scrive nel segment dati all'indirizzo specificato in $a0
sbrk	9	Numero di byte da allocare in $a0	Indirizzo del primo dei byte allocati in $v0	Accresce il segmento dati allocanto un numero di byte specificato in $a0, restituisce in $v0 l'indirizzo del primo di questi nuovi byte
exit	10	nessuno	nessuno	Termina l'esecuzione

133

System Calls «Apocrife»
(fornite dalla macchina virtuale di MIPSweb)

Syscall	Codice	Argomenti	Valore di ritorno	Descrizione
Time	30	nessuno	32 bit meno significativi del system time in $a0, 32 bit più significativi del system time in $a1	Il system time è rappresentato nel formato Unix Epoch time, cioè il numero di millisecondi trascorsi dal 1 Gennaio 1970
random int	41	Id del generatore pseudo-random in $a0	Prossimo numero pseudo random in $a0	Ad ogni chiamata restituisce un numero intero in una sequenza pseudo-random
random in range	42	Id del generatore pseudo-random in $a0, massimo intero generabile in $a1	Prossimo numero pseudo random in $a0	Ad ogni chiamata restituisce un numero intero in una sequenza pseudo-random, ogni numero sarà compreso tra 0 e il massimo passato in $a1
MessageDialog	55	Indirizzo della stringa da stampare in $a0, intero corrispondente al tipo di messaggio in $a1		Mostra una finestra di dialogo con un messaggio dato dalla stringa passata in $a0. Viene anche mostrata una icona che dipende dal tipo di messaggio passato in $a1: errore (0), info, (1), warning (2), domanda(3)
InputDialogInt	51	Indirizzo della stringa da stampare in $a0	Intero letto in $a0, stato in $a1	

134

Un primo esempio di programma *interattivo* (con I/O)

```
.data

msg: .asciiz "Immetti un numero che te lo raddoppio: "

.text

li $v0 4        #
la $a0 msg      # print( msg )
syscall         #

li $v0 5        #
syscall         # $s0 = read_number()
move $s0 $v0    #

mul $s1 $s0 2   # $s1 = $s0 * 2

li $v0 1        #
move $a0 $s1    # print( $s1 )
syscall         #

li $v0 10       #
syscall         # exit
```

135