



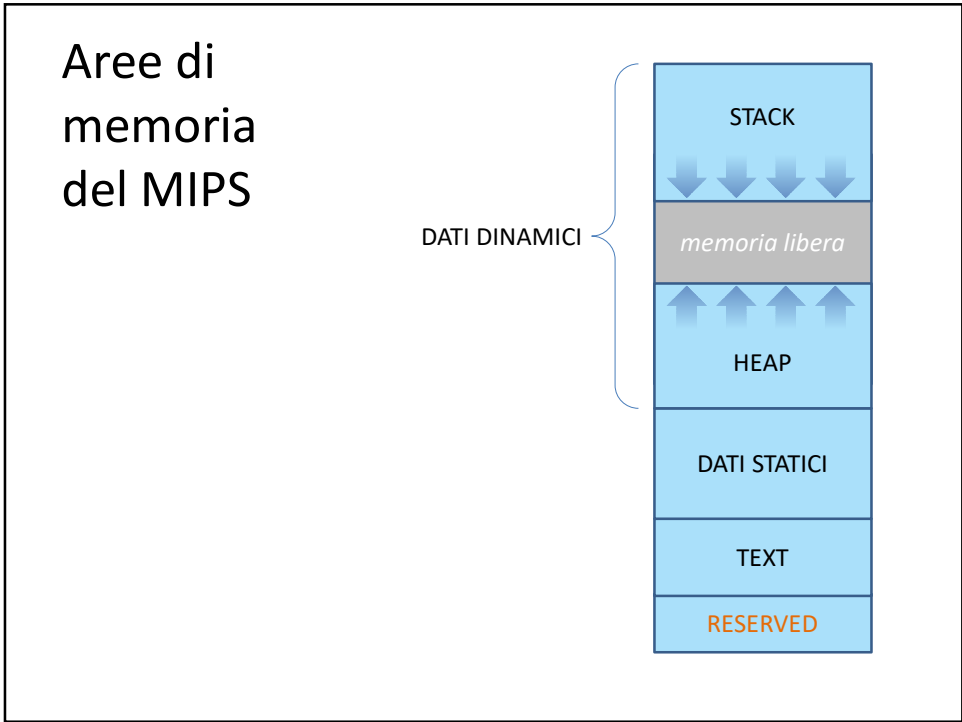
Università degli Studi di Milano
Dipartimento di Informatica “Giovanni Degli Antoni”
Corso di Laurea Triennale in Informatica

Architettura degli Elaboratori II

Laboratorio

Allocazione dinamica della memoria

1



3

Allocazione statica vs. allocazione dinamica

STATICA

- Avviene a **compile-time**
- Non viene mai deallocata (fino al termine dell'esecuzione)
- Dimensione fissa
- Ad alto livello: usata per variabili globali, array globali di dimensione fissa

DINAMICA

- Avviene a **run-time**
- Può essere dellocata a **run-time**
- La dimensione può variare

6

Allocazione **dinamica**: heap VS stack

STACK

- Allocazione e deallocazione: automatica, FIFO
- Ad alto livello: usato per ospitare variabili locali, array locali con dimensione fissa

HEAP

- Allocazione e deallocazione: a discrezione del programmatore (o del garbage collector, se c'è)
- Ad alto livello: usato per array (locali, o globali) di dimensione mutevole e/o non nota staticamente, e altri dati di cui si vuole controllare la persistenza in memoria

7

Come avviene l'allocazione?

1. Il programma richiede al sistema operativo di allocare un certo numero di byte in memoria tramite una syscall

2. Il sistema operativo verifica l'effettiva disponibilità dello spazio in memoria e, se possibile, alloca lo spazio richiesto e restituisce al programma il base address dell'area di memoria allocata

3. In caso di memoria non allocabile, di norma, il sistema operativo restituirà un codice di errore (per es, un numero negativo) al posto del base address permettendoci di gestire il problema

```
graph TD; A[my_prog.asm] -- SYSCALL --> B[OS]; B -- "Base address: 0x10040000" --> A;
```

In MARS: la richiesta di allocare più spazio di quanto disponibile, genera un'eccezione a run time

8

System Calls di MIPS				
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	nessuno	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 allocando 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

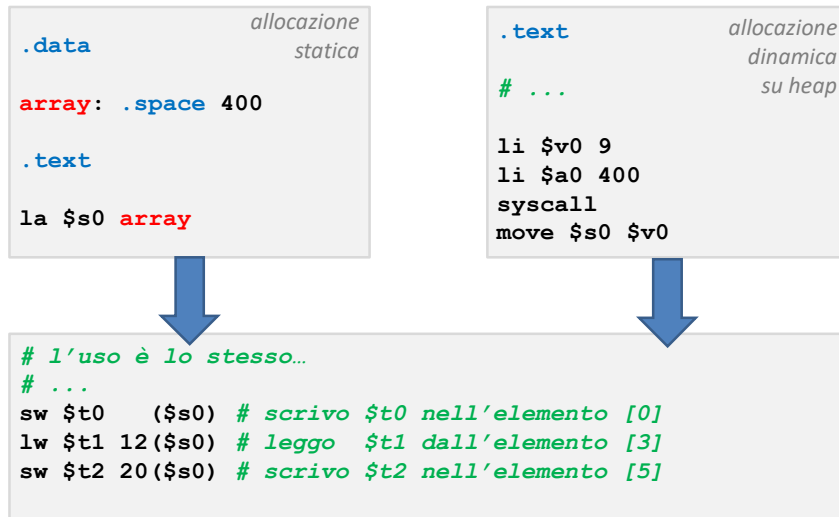
9

Marco Tarini

Unvierstità degli Studi di Milano

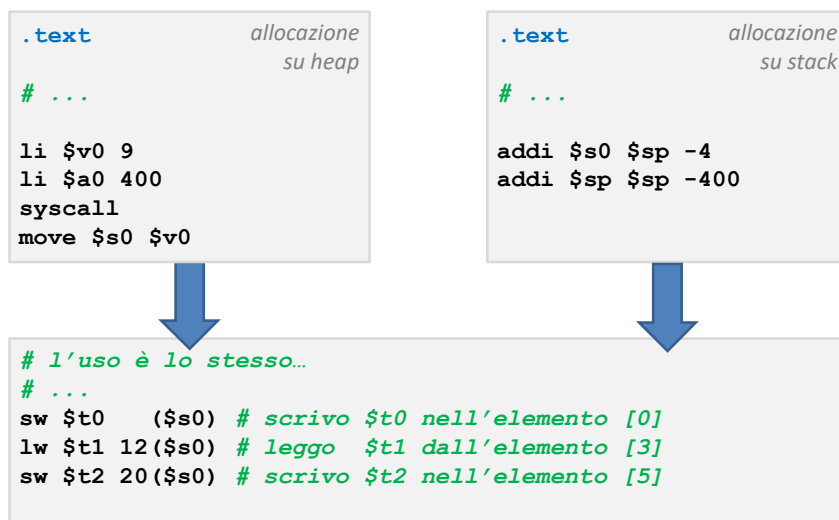
3

Allocazione statica VS dinamica di un vettore di 100 words




10

Allocazione su HEAP vs STACK di un vettore di 100 words



11

Streterie di allocaione dello Heap a confronto



MIPS	C	C++	C#, Java...	Go
sbrk +XXX	malloc, calloc	new (+STL)	new	new, make, decisione del compilatore
sbrk -XXX	free	delete (+STL)	<i>garbage collector</i>	<i>garbage collector</i>

17

SBRK (system call in MIPS) vs. malloc (funzione di lib standard in C)

Similitudini:

- entrambe prendono in input il numero X di byte da allocare
- entrambe restituiscono l'indirizzo del pezzo di memoria allocato
- entrambe allocano (riservano, preservano da altri usi) un nuovo brano di memoria di X byte nello HEAP

Differenze:

- La SBRK si occupa solo di «spostare» il puntatore alla fine dello heap più avanti, aumentando la lunghezza dello heap di quanto richiesto (e quindi lo spazio di memoria dinamica disponibile al nostro programma)
- La malloc è molto più complicata: cerca un brano di memoria contigua disponibile nello heap (non necessariamente in cima, ma anche in mezzo – come un brano lasciato libero da una free precedente).
- La malloc può, oppure no, allungare lo heap. Se lo fa, lo fa invocando (a basso livello!) una SBRK.

18

SBRK negativo vs. free (funzione di lib standard in C)

- La SBRK con input negativo prende in input l'opposto del numero X di byte da deallocare, e non restituisce output
- La SBRK si occupa solo di «spostare» il puntatore alla fine dello heap più indietro, diminuendo la lunghezza dello heap di tanto quanto richiesto (e quindi lo spazio di memoria dinamica disponibile al nostro programma)
- La free è molto più complicata: marca come libero un brano di memoria sullo heap (non necessariamente quello in cima).
- Se è quello in cima, allora decrementa lo heap (con una SBRK!)
- Altrimenti, si limita a creare uno spazio vuoto all'interno dello heap, che potrà essere riallocato in seguito da nuove malloc (nota: inoltre, gli spazi vuoti contigui possono dover essere fusi)

19

SBRK (segment break)

La syscall utilizzata per allocare spazio dinamicamente sullo heap è la **SBRK** (codice 9).

- Input: \$a0 <- numero di byte da allocare
- Output: \$v0 <- base address dell'area di memoria allocata

```
.text
.globl main

main:
    li $v0 9 # codice per SBRK
    li $a0 100 # chiedo di allocare 100 bytes
    syscall

    # $v0 ora contiene il base address dell'area di memoria allocata

    # carico il valore 5 nella prima parola di memoria allocata
    li $t0 5
    sw $t0 0($v0)

    # carico 6 nella seconda
    li $t0 6
    sw $t0 4($v0)
```

21

System Calls "Apocrife"				
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	