Laboratorio di Architettura degli Elaboratori II

Corso di laurea triennale in Informatica
Università degli Studi di Milano, A.A. 2019-2020

Appello del 27 Gennaio 2020

- L'esame ha una durata di 3 ore ed è composto da 3 esercizi.
- È possibile consultare il libro di testo, appunti e la documentazione di MARS.
- È proibito l'accesso ad Internet con qualsiasi mezzo.
- Verranno corretti solo gli esercizi che non generano errori in compilazione ed esecuzione.

1 Grandi e piccoli

Upload: `funzione1.asm`

Si implementi la funzione `funzione1` (invocabile da altri file sorgenti) che riceva in input, nell'ordine:

- il base address di un array di interi `A`;
- un intero `k` che indichi il numero di elementi contenuti in `A`.

Immaginiamo che i `k` interi in `A` siano divisi in due gruppi. Il primo gruppo comprende la metà più uno dei numeri più grandi dell'array (la "maggioranza dei grandi"). Sia `M` la somma dei numeri nel primo gruppo. Il secondo gruppo comprende i restanti più piccoli (la "minoranza dei piccoli"). Sia `m` la somma dei numeri nel secondo gruppo.

Se `A` contiene almeno `3` elementi (`k>=3`), `funzione1` restituisce un intero pari a `M - m`. In caso contrario restituisce `-1`.

Nota: non è necessario lasciare l'array inalterato dopo l'esecuzione

1.1 Esempi

- Con `A=[45,3]`, `k=2` la funzione restituisce `-1`.
- Con `A=[1,2,3,4,5,6,7]`, `k=7`, il primo gruppo è dato da `[4,5,6,7]` da cui `M=22`. Il secondo gruppo è dato da `[1,2,3]` da cui `m=6`. La funzione ritornerà `22-6=16`.
- Con `A=[45,3,20,7,11,18,9,57,32,44]`, `k=10`, il primo gruppo è dato da `[18,20,32,44,45,57]` da cui `M=216`. Il secondo gruppo è dato da `[3,7,9,11]` da cui `m=30`. La funzione ritornerà `216-30=186`.

2 Cercasi funzione2

Upload: `main.asm, funzione2.asm`

Il seguente frammento di codice assembly (file `main.txt` allegato) presenta due problemi. Il primo è che non ci sono commenti che possano facilitare la comprensione delle operazioni svolte. Il secondo è che il file contenente l'implementazione della funzione `funzione2` non è stato fornito. Si ponga rimedio ai due problemi.

```
    .data
msg_1:  .asciiz " è la più lunga\n"
msg_2:  .asciiz " è la più corta\n"
msg_3:  .asciiz "Stessa lunghezza!\n"
buff_1: .space 1024
buff_2: .space 1024

    .text
    .globl main
main:
    li $v0 8
    li $a1 1024
    la $a0 buff_1
    syscall

    li $v0 8
    li $a1 1024
    la $a0 buff_2
    syscall

    la $a0 buff_1
    la $a1 buff_2
    jal funzione2

    beqz $v0 peq
    bgez $v0 c1
c2:
    la $t0 msg_2
    la $t1 msg_1
    j pdiff
c1:
    la $t0 msg_1
    la $t1 msg_2
pdiff:
    li $v0 4
    la $a0 buff_1
```

```
    syscall
    li $v0 4
    move $a0 $t0
    syscall

    li $v0 4
    la $a0 buff_2
    syscall
    li $v0 4
    move $a0 $t1
    syscall
    j end
peq:
    li $v0 4
    la $a0 msg_3
    syscall
end:
    li $v0 10
    syscall
```

3 Compara stringhe

Upload: `funzione3.asm`

Si implementi la funzione `funzione3` che, date due stringhe `s1` e `s2` della stessa lunghezza restituisca:

- -3 se le stringhe `s1` e `s2` passate corrispondono alla stessa variabile (hanno lo stesso indirizzo);
- `-2` se almeno una delle due stringhe non è composta unicamente da lettere minuscole `[a, b, c, ..., z]`;
- 0 se le stringhe hanno lo stesso valore (ad esempio, `s1="treno"` e `s2="treno"`);
- 1 se `s1` precede alfabeticamente `s2` (ad esempio, `s1="treni"` e 's2="treno");
- -1 in ogni altro caso.

Nota: nello svolgimento dell'esercizio si assuma pure che il chiamante passi sempre due stringhe della stessa lunghezza.

4 Tabella dei codici ASCII

#	Ch	#	Ch	#	Ch	#	Ch
0	NUL (null)	32	SPACE	64	@	96	‘
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	“	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	’	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL