[Sessione 8] Simulazione d'esame

[s8.1] Write-a

nome del file sorgente: writea.asm

(A) Utilizzando le direttive assembly, si scriva un segmento dati che contenga:

- una variabile intera *a*, inizializzata con un valore a piacere;
- una stringa *s* con valore "Fine";
- un array *V* di `5` elementi, dove ogni elemento è inizializzato a 1.

(B) Si scriva poi un segmento testo che:

- per *i* che va da *0* a *4*, memorizzi il valore *a+i* nella posizione *i* dell'array;
- memorizzi nella variabile *a* l'indirizzo del penultimo elemento dell'array;
- stampi la stringa *s*.

[s8.2] BranchY

nome del file sorgente: branchy.asm

(A) Si implementi, senza fare uso di pseudo-istruzioni, una procedura chiamata "*branchy*" così definita:

- Input: due valori interi *a* e *b*, due indirizzi del segmento testo *T1* e *T2*.
- Output: un indirizzo del segmento testo.

La procedura restituisce un indirizzo determinato da queste condizioni:

- se *a>b* viene restituito *T1*;
- se *a<b* viene restituito *T2*;
- (B) se *a=b* viene restituito l'indirizzo dell'istruzione che, nel chiamante, precede la chiamata alla procedura.

[s8.3] Emerge2

nome del file sorgente: emerge2.asm

Si implementi una procedura chiamata "*emerge2*" così definita:

- Input: due interi *i* e *j*, un array *V* (base address) e un intero *k* che rappresenta il numero di elementi in *V*.
- Output: un valore booleano.

Si assuma che $i \leq j$.

(A) In caso in cui i o j non corrispondano ad indici ammissibili di V, la procedura termina immediatamente restituendo `0`.

(B) In caso contrario, la procedura modifica l'ordine degli elementi in V in modo tale che gli elementi in posizione i e j vadano ad occpuare la prima e seconda posizione, rispettivamente. Tutti i restanti elementi di V vengono shiftati verso il fondo. Al ritorno di una chiamata a *emerge2*, l'elemento originariamente in posizione i occuperà la posizione `0` (primo elemento); l'elemento originariamente in posizione j, se j è diverso da i, occuperà la posizione `1` (secondo elemento); lo shift verso il fondo di ogni elemento che occupava una posizione diversa da i e da j garantirà che nessun elemento di V venga sovrascritto e perso. Dopo aver effettuato questa operazione sull'array, la procedura termina restituendo `1`.

Esempio (indici errati):

Data questa configurazione di input:

```
i=23,j=-4, V=[1,2,3,4,5,6,7], k=7
```

Al ritorno dalla procedura si avrà:

```
V=[1,2,3,4,5,6,7]
$v0=0
```

Esempio (indici validi ma uguali):

Data questa configurazione di input:

```
i=5,j=5, V=[1,2,3,4,5,6,7], k=7
```

Al ritorno dalla procedura si avrà:

```
V=[6,1,2,3,4,5,7]
$v0=1
```

Esempio (indici validi e diversi):

Data questa configurazione di input:

```
i=3,j=5, V=[1,2,3,4,5,6,7], k=7
```

Al ritorno dalla procedura si avrà:

```
V=[4,6,1,2,3,5,7]
$v0=1
```