esercizio n. 2 – linguaggio macchina

Si deve tradurre in linguaggio macchina simbolico (linguaggio assemblatore) 68000 il programma C (programma principale *main* e funzione *mkelems*) riportato sotto. Non si tenti di accorpare od ottimizzare insieme istruzioni C indipendenti. La memoria ha indirizzi da **32 bit**, è indirizzabile per byte e i puntatori sono da **32 bit**. Si facciano le ipotesi seguenti:

- i parametri delle le funzioni sono passati sulla pila in ordine inverso di elencazione
- i valori restituiti dalle funzioni ai chiamanti rispettivi sono passati sulla pila, sovrascrivendo il primo parametro passato o nello spazio libero lasciato appositamente
- le variabili locali vengono impilate in ordine di elencazione
- le funzioni salvano sempre i registri che utilizzano (tranne il *main*)

Si chiede di svolgere i tre punti seguenti:

1. **Scrivere** l'aera di attivazione della funzione *mkelems*, secondo il modello 68000.
2. **Scrivere,** in linguaggio macchina 68000, il codice di *mkelems*, coerentemente con le specifiche e con le risposte ai punti precedenti (il numero di righe non è significativo e le prime righe sono già date).
3. Si immagini di simulare il funzionamento del programma, fino a quando la seconda chiamata a *mkelems* (la prima è quella che figura in *main*) si trova subito dopo lo **statement 1**, sapendo che la prima chiamata alla funzione *malloc* restituisce **2500** e la seconda **2668**. **Rispondere** ai due sotto-punti seguenti:
   a) **indicare** lo stato della pila**,** scrivendo i contenuti numerici delle celle della pila, questi ultimi se sono precisabili; si scriva anche il nome simbolico delle celle di pila di cui si dà il contenuto numerico, utilizzando la colonna della domanda 1
   b) **indicare** i valori contenuti nei registri FP e SP**,** subito dopo la LINK della 2ª chiamata a *mkelems*

```
#define LEN  = 5
#define NULL = 0
typedef struct ELEM_t {
      struct ELEM_t * next;
} ELEM_t;

/* funzione malloc – allocazione dinamica di memoria            */
void * malloc (int)    /* testata della funzione                */
```

```
/* variabili globali                                            */
ELEM_t * head = NULL;

ELEM_t * mkelems (ELEM_t * old, int n) {
   ELEM_t * new;
   if (n == 0) {
      return old;
   } else {
      new = malloc (sizeof (ELEM_t));
      new->next = old;              /* statement 1 – concat.    */
      return mkelems (new, n – 1);  /* chiamata ricorsiva       */
   } /* if */
} /* mkelems */

void main ( ) {
   head = mkelems (NULL, LEN);
} /* main */
```

indirizzo	**area di attivazione** (simbolica) di *mkelems* (domanda **1**)		**contenuto** (numerico) della **pila** subito **dopo** avere eseguito lo **statement 1** nella **seconda** chiamata a *mkelems* (domanda **3**)
3008	*reg. D0 e A0*	*- 8 byte*	*4 e 2500*
3012	*new*	*- 4 byte*	*2668 (punt a elem testa lista)*
3016	*FP precedente*	*- 4 byte*	*3044 (ind campo FP di 1ª mkelems)*
3020	*indirizzo di rientro*	*- 4 byte*	*indirizzo di rientro a mkelems*
3024	*old*	*- 4 byte*	*2500*
3028	*n e valusc*	*- 4 byte*	*4 (LEN – 1)*
3032	*reg. D0 e A0*	*- 8 byte*	*non precisabili (sono di main)*
3040	*new*	*- 4 byte*	*2500 (punt a elem fondo lista)*
3044	*FP precedente*	*- 4 byte*	*3060 (ind campo FP di main)*
3048	*indirizzo di rientro*	*- 4 byte*	*indirizzo di rientro a main*
3052	*old*	*- 4 byte*	*NULL*
3056	*n e valusc*	*- 4 byte*	*5 (LEN)*
3060	**FP precedente a main**		NULL (non punta a niente)

	valore (numerico) subito **dopo** avere eseguita l'istruzione LINK nella **seconda** chiamata a *mkelems* (domanda **3**)
registro FP	*3016*
registro SP	*3008*

In rosso è messa in evidenza l'area di attivazione di mkelems, data in risposta alla domanda 1; il resto (in nero) è lo stato della pila, numerico e simbolico, dato in risposta alla domanda 3.

codice 68000 di *mkelems* (domanda **2**) *– NON OTTIMIZZATO*			
MKELEMS:	**LINK**	FP, #-4	// AGGIUNGERE INGOMBRO VARLOC
N:	**EQU**	12	// AGGIUNGERE SPIAZZ.
OLD:	**EQU**	8	// AGGIUNGERE SPIAZZ.
NEW:	**EQU**	-4	// AGGIUNGERE SPIAZZ.
	MOVEM.L	D0/A0, -(SP)	// salva D0 e A0 in pila
IF:	**MOVE.L**	N(FP), D0	// carica *n*
	CMPI.L	#0, D0	// confronta
	BNE	ELSE	// se != 0 va' a ELSE
THEN:	**MOVE.L**	OLD(FP), N(FP)	// sovrascrivi valusc
	BRA	ENDIF	// va' a fine IF
ELSE:	**MOVE.L**	#4, -(SP)	// impila sizeof (ELEM_t) = 4
	BSR	MALLOC	// chiama *malloc*
	MOVE.L	(SP)+, NEW(FP)	// spila valusc e aggiorna *new*
	MOVEA.L	NEW(FP), A0	*// new in A0 per aggiornam. next*
	MOVE.L	OLD(FP), (A0)	// aggiorna *next*
	MOVE.L	N(FP), D0	// carica *n*
	SUBI.L	#1, D0	// decrementa
	MOVE.L	D0, -(SP)	// impila *n – 1* come 2° arg
	MOVE.L	NEW(FP), -(SP)	// impila *new* come 1° arg
	BSR	MKELEMS	// chiama *mkelems* (ricorsivamente)
	ADDA.L	#4, SP	// abbandona *old*
	MOVE.L	(SP)+, N(FP)	// spila e sovrascrivi valusc
ENDIF:	**MOVEM.L**	(SP)+, D0/A0	// ripristina D0 e A0 da pila
	UNLK	FP	// scollega
	RFS		// rientra

Osservazioni: il registro A0 è usato come puntatore; la funzione mkelems è leggermente ottimizzata sfruttando l'ortogonalità di MOVE.