POLITECNICO DI MILANO

# Esercizi
# Assembly

Alessandro A. Nacci
alessandro.nacci@polimi.it

## ACSO
2014/2014

# Esercizio 1

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*

SOLUZIONI ONLINE

# Esercizio 1

```
#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 */
```

# Esercizio 2

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

- i parametri delle 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 sono impilate in ordine di elencazione
- le funzioni salvano sempre i registri che utilizzano (tranne *main*)

**Si chiede** di svolgere i tre punti seguenti:

1. **Si descriva** l'area di attivazione della funzione *rec* e quella della funzione *funz*, secondo il modello 68000, nelle due tabelle predisposte.
2. **Si traduca** in linguaggio macchina 68000 il codice di *rec* e quello di *main*, coerentemente con le specifiche e le risposte ai punti precedenti, usando le due tabelle predisposte.
3. **Si immagini** di simulare il funzionamento del programma, con l'ipotesi **MAX = 1**, fino a quando la funzione *funz* ha appena messo il valore di uscita nella sua area di attivazione. **Si indichi** lo stato della pila, e precisamente il contenuto delle aree di attivazione delle funzioni *rec* e *funz* riferite a tale punto di esecuzione, riportando sia il contenuto simbolico delle celle della pila sia quello numerico (se precisabile), nella tabella predisposta.

SOLUZIONI ONLINE

# Esercizio 2

```
#define MAX  = 3

/* variabili globali */
int a = 5;
int b = 10;
int h;

/* funzioni */
int funz (int p, int q) {
   int k;  /* variabile locale */
   k = p * q;
   return k - q;
} /* funz */

int rec (int n, int x, int y) {
   if (n == 0) {
      return x + y;
   } else {
      /* chiamata ricorsiva */
      return funz (x - y, rec (n - 1, x, y));
   } /* if */
} /* rec */

/* programma principale */
void main ( ) {
   h = rec (MAX, a, b);
} /* main */
```

# Esercizio 3

Si deve tradurre in linguaggio macchina simbolico (linguaggio assemblatore) 68000 il programma (main e funzione *funz*) riportato qui sotto. Nel tradurre non si tenti di accorpare od ottimizzare insieme istruzioni C indipendenti.

La memoria ha parole da **16 bit**, indirizzi da **32 bit** ed è indirizzabile per byte. Le variabili intere sono da **16 bit**. Ulteriori specifiche al problema e le convenzioni da adottare nella traduzione sono le seguenti:

- i parametri di tutte 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 dei parametri passati o nello spazio libero opportunamente lasciato
- le variabili locali vengono impilate in ordine di elencazione
- le funzioni (tranne *main*) devono sempre salvare i registri che utilizzano

**Si chiede** di svolgere i **tre** punti seguenti:

1. **Si descriva** la struttura della memoria delle variabili globali e l'area di attivazione della funzione *funz*, secondo il modello 68000, nelle tabelle predisposte, indicando gli spiazzamenti rispetto a FP.
2. **Si dichiarino** in linguaggio macchina 68000 le variabili globali e **si scriva** il codice macchina della funzione *main*, coerentemente con le specifiche e le risposte precedenti, usando le tabelle predisposte.
3. **Si scriva** in linguaggio macchina 68000 il codice macchina della funzione *funz*, coerentemente con le specifiche e le risposte ai punti precedenti, usando le tabelle predisposte.

SOLUZIONI ONLINE

# Esercizio 3

**programma in linguaggio C**

```
#define N = 5

/* variabili globali */

int dati [N];
int i;
int r = 0;

/* programma main (alcune parti sono volutamente omesse) */

void main ( ) {
   i = N - 1;
   do {
      if (dati [i] < dati [i - 1]) {
         r = r + funz (dati [i], i);
      }  /* fine if */
      i--;
   while (i >= 1);
   }  /* fine do */
} /* fine main */

/* funzione funz */

int funz (int a, int b) {
   return (a + r) * b;
} /* fine funzione */
```

# Esercizio 3 (parte II)

Si supponga che la memoria abbia parole da **16 bit** e sia indirizzata per **byte**. Si svolgano i punti seguenti:

a) Nella tabella sotto, **si riportino** gli indirizzi dove collocare dati e istruzioni. Si consideri che ogn istruzione ha sempre una parola di codice operativo e, se serve, una o più parole aggiuntive. Si tenga conto che lo spiazzamento sia in istruzioni che manipolano dati sia in istruzioni di salto è sempre da **16 bit** e che indirizzi e costanti sono **corti** (16 bit) o **lunghi** (32 bit) come specifica l'istruzione.

| | | | # parole | # byte | indirizzo (in decimale) |
|---|---|---|---|---|---|
| | **ORG** | 5000 | | | |
| ROW: | **EQU** | 2 | | | |
| COL: | **EQU** | 3 | | | |
| MAT: | **DS.L** | 6 | | | |
| SUM: | **DC.W** | 0 | | | |
| INIZ: | **MOVEA.L** | #MAT.L, A0 | | | |
| | **CLR.L** | D1 | | | |
| OUTER: | **CMPI.L** | #ROW.L, D1 | | | |
| | **BGE** | FINISH | | | |
| | **CLR.W** | D2 | | | |
| INNER: | **ADD.W** | (A0, D2), SUM.L | | | |
| | **ADDI.W** | #1.W, D2 | | | |
| | **CMPI.W** | #COL.W, D2 | | | |
| | **BLT** | INNER | | | |
| | **ADDI.L** | #ROW.L, A0 | | | |
| | **ADDI.L** | #1.L, D1 | | | |
| | **BRA** | OUTER | | | |
| FINISH: | **END** | INIZ | | | |

| *Tabella dei simboli* | |
|---|---|
| *nome simbolo* | *valore numerico (in decimale)* |
| ROW | |
| COL | |
| MAT | |
| SUM | |
| INIZ | |
| OUTER | |
| INNER | |
| FINISH | |

SOLUZIONI ONLINE

# Esercizio 3 (parte II)

c) **Si dica** quanto vale in decimale lo spiazzamento codificato nell'istruzione BGE:

d) **Si dica** quanto vale in decimale lo spiazzamento codificato nell'istruzione BLT:

e) **Si dica** quale valore (decimale) verrà caricato nel registro PC al lancio del programma:

SOLUZIONI ONLINE

# Esercizio 4

Si deve tradurre in linguaggio macchina simbolico (linguaggio assemblatore) 68000 il programma (*main* e funzione *funz*) riportato qui sotto. Nel tradurre non si tenti di accorpare od ottimizzare insieme istruzioni C indipendenti.

La memoria ha parole da **16 bit**, indirizzi da **32 bit** ed è indirizzabile per byte. Le variabili intere sono da **32 bit**. Ulteriori specifiche al problema e le convenzioni da adottare nella traduzione sono le seguenti:

- i parametri di tutte 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 dei parametri passati o nello spazio libero opportunamente lasciato
- le variabili locali vengono impilate in ordine di elencazione
- le funzioni (tranne *main*) devono sempre salvare i registri che utilizzano

**Si chiede** di svolgere i **tre** punti seguenti:

1. **Si descriva** la struttura della memoria delle variabili globali e l'area di attivazione della funzione *funz*, secondo il modello 68000, nelle tabelle predisposte, indicando gli spiazzamenti rispetto a FP.
2. **Si dichiarino** in linguaggio macchina 68000 le variabili globali e **si scriva** il codice macchina della funzione *main*, coerentemente con le specifiche e le risposte precedenti, usando le tabelle predisposte.
3. **Si scriva** in linguaggio macchina 68000 il codice macchina della funzione *funz*, coerentemente con le specifiche e le risposte ai punti precedenti, usando le tabelle predisposte.

SOLUZIONI ONLINE

# Esercizio 4

**programma in linguaggio C**

```
/* costanti */
#define  N  6
/* variabili globali */
int x = N;
int y;
int *p;
/* funzione funz */
int funz (int a, int * q) {
   int b;
   if (*q == a) {
      return a;
   } else {
      b = funz (a - 1, q);
      return b;
   } /* if */
} /* fine funz */
/* programma main */
void main ( ) {
   p = &x;
   y = funz (N, p);
} /* fine main */
```

SOLUZIONI ONLINE

# Esercizio 4 (parte II)

Si consideri la codifica della funzione **F1** sotto riportata che viene assemblata come un modulo a sé stante e si supponga che la memoria abbia parole da **16 bit** e sia indirizzata per **byte**. Si svolgano i punti seguenti:

a) Nella tabella sotto, **si riportino** gli indirizzi dove collocare dati e istruzioni. Si noti che la riga della prima istruzione di F1 è già stata compilata e i valori inseriti devono essere tenuti in conto per calcolare l'indirizzo successivo. Si consideri che ogni istruzione ha sempre una parola di codice operativo e, se serve, una o più parole aggiuntive. Si tenga conto che lo spiazzamento sia in istruzioni che manipolano dati sia in istruzioni di salto è sempre da **16 bit** e che indirizzi e costanti sono **corti** (16 bit) o **lunghi** (32 bit) come specifica l'istruzione.

SOLUZIONI ONLINE

# Esercizio 4 (parte II)

| | | | # parole | # byte | indirizzo (in decimale) |
|---|---|---|---|---|---|
| | **ORG** | 5000 | | | |
| F1: | **LINK** | FP, #-2.W | | | |
| X: | **EQU** | +8 | | | |
| Y: | **EQU** | +3 | | | |
| RES: | **EQU** | -2 | | | |
| | **MOVEM** | D0-D1,-(SP) | | | |
| | **MOVE** | #0.W, RES(FP) | | | |
| WHILE: | **MOVE** | X(FP), D0 | | | |
| | **CMPI** | #0.W, D1 | | | |
| | **BLT** | END_WHILE | | | |
| | **MOVE** | Y(FP), D1 | | | |
| | **SUB** | D0, D1 | | | |
| | **MOVE** | D1, -(SP) | | | |
| | **BSR** | F2 | | | |
| | **MOVE** | (SP)+, RES(FP) | | | |
| | **ADDI** | #-1.W, D0 | | | |
| | **MOVE** | D0, X(FP) | | | |
| | **BRA** | WHILE | | | |
| END_WHILE: | **MOVE** | RES(FP), Y(FP) | | | |
| | **UNLINK** | FP | | | |
| | **RTS** | | | | |

SOLUZIONI ONLINE

# Esercizio 4 (parte II)

b) Nella tabella data sotto, **si riportino** i simboli e – dove possibile – i rispettivi valori numerici.

| *Tabella dei simboli* | |
|---|---|
| *nome simbolo* | *valore numerico (in decimale)* |
| F1 | |
| X | |
| Y | |
| RES | |
| WHILE | |
| END_WHILE | |
| F2 | |

# Alla prossima lezione

alessandro.nacci@polimi.it