

# Documentazione acm-s

di Angelo Theodorou

L'acm-s è una versione semplificata del processore immaginario acm, descritto nel testo di PETER BISHOP “*L'Informatica*”, edito dalla Jackson Libri.

Il programma in questione è un simulatore di tale processore, ed è stato implementato in C.

## Interfaccia Utente

All'avvio del programma viene presentato un comodo menù a 6 voci che mette a disposizione dell'utente le seguenti azioni:

### 1. LOAD

Serve a caricare un file nella memoria RAM dell'elaboratore simulato.

Conseguentemente viene stampato all'utente un messaggio che lo invita a digitare il nome del file da caricare.

### 2. DUMP

La funzione di questo comando è quella di mostrare il contenuto della memoria dell'elaboratore in quell'istante. Si richiede un indirizzo di partenza ed uno di arrivo, dopodiché viene mostrato a video il dump della RAM.

### 3. RUN

Il programma caricato è messo in esecuzione fino a quando non terminerà.

### 4. TRACE

Vengono stampati i contenuti di tutti i registri, quindi l'istruzione puntata dal Program Counter è messa in esecuzione.

### 5. RESET

Azzerare tutti i registri del processore lasciando però intatto il contenuto della RAM.

### 6. EXIT

Termina l'esecuzione del simulatore e torna al sistema operativo.

## Programmi annessi

Il programma è distribuito con la seguente fornitura di programmi in linguaggio macchina, testati nella loro funzionalità:

- add - Stampa la somma di due numeri inseriti in input.
- equal - Avvalendosi di un salto condizionale, restituisce 1 se i due numeri in input sono uguali, altrimenti 0.
- countdown - Richiede un numero di partenza all'utente, poi stampa sull'output tutti i numeri a partire da quello fino allo zero.
- neg - Richiede un numero positivo e lo complementa a 2 per ottenere il corrispondente negativo. (L'operazione viene realizzata mediante l'istruzione di complementazione di un registro, che non è ancora implementata nel simulatore)
- over - Esegue addizioni e sottrazioni sull'accumulatore per testare le condizioni di overflow.
- over2 - Incrementa e decrementa l'accumulatore per testare le condizioni di overflow.
- sub - Stampa la differenza di due numeri inseriti in input.
- array - Memorizza uno dopo l'altro in memoria i caratteri inseriti in input, si ferma se l'utente inserisce 0xff. Il programma si avvale dell'accesso con modo di indirizzamento indiretto ad una locazione di memoria che funge da puntatore al prossimo elemento dell'array.
- mul - stampa il prodotto di due numeri inseriti in input.

## Implementazione dell'hardware

### Registri:

- `unsigned char mem[MAXMEM];`

E' un array di caratteri, ovvero variabili lunghe 8 bit. Ogni coppia di elementi dell'array contiene quindi una parola di macchina (la CPU è a 16 bit).

- `unsigned short int pc;`

Questo è il Program Counter, ovvero il puntatore all'istruzione da eseguire.

E' un intero senza segno per far sì che il processore possa accedere a 64k di RAM, il massimo per un processore con un Address Bus di 16 bit.

- `unsigned short int reg_instr;`

Il registro di istruzione contiene una copia dell'istruzione in quel momento puntata dal PC.

- `unsigned short int reg_addr_mem;`

Il registro di indirizzo memoria contiene l'indirizzo del dato da elaborare in quel momento.

- `short int reg_data_mem;`

Il registro dati memoria contiene proprio il dato sul quale si vuole operare.

- `short int accum;`

L'accumulatore è il registro interno del processore sul quale lavora l'ALU.

- `int zero, carry, negative, overflow;`

Questi sono i codici di condizione associati alle operazioni svolte.

- `char input, output;`

I due registri a 8 bit di input ed output rappresentano l'unica interazione tra l'utente del programma e la macchina.

### Istruzioni:

→ M – modo di indirizzamento :

1. operando immediato
2. indirizzo assoluto
3. indirizzo indiretto

→ R – identificatore di registro

1. accumulatore

→ P – unità periferica

1. terminale

→ XX – indirizzo relativo di eliminazione di salto

→ Effetti sui codici di condizione:

S - impostazione (diventa 1)

C - azzeramento

N - nessun effetto

D – condizionato dal risultato

→ Codici di condizione

Z - zero

N - negativo

C – riporto/prestito

V - tracimazione

<i>Istruzione</i>	<i>Interpretazione</i>	<i>Z</i>	<i>N</i>	<i>C</i>	<i>V</i>
<i>Gruppo di indirizzamento memoria</i>					
0x11RM	Carica la parola del dato nel registro	D	D	N	N
0x12RM	Memorizza la parola del registro in memoria	D	D	N	N
0x13RM	Aggiunge la parola del dato al registro	D	D	D	D
0x14RM	Aggiunge la parola del dato ed il bit di riporto al registro	D	D	D	D
0x15RM	Sottrae la parola del dato del registro	D	D	D	D
0x17RM	Fa un AND della parola del dato con il registro	D	D	C	C
0x18RM	Fa un OR della parola del dato con il registro	D	D	C	C
0x19RM	Fa un OR esclusivo della parola del dato con il registro	D	D	C	C
0x1aRM	Confronta il registro con la parola del dato	D	D	D	D
<i>Gruppo di manipolazione di registro</i>					

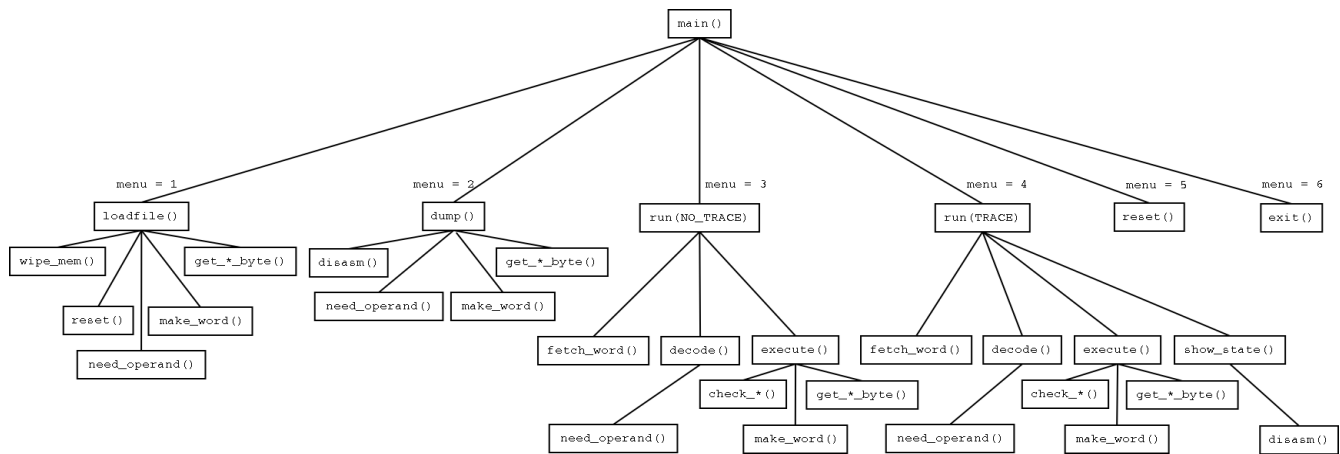
<b>Istruzione</b>	<b>Interpretazione</b>	<b>Z</b>	<b>N</b>	<b>C</b>	<b>V</b>
0x01R0	Azzera registro	S	C	C	C
0x02R0	Incrementa il registro di 1	D	D	D	D
0x03R0	Decrementa il registro di 1	D	D	D	D
0x09R0	Nega il registro (operazione NOT)	D	D	C	C
<b>Salti incondizionati</b>					
0x5100	Salto incondizionato ad un indirizzo specificato	N	N	N	N
<b>Salti diramazionali</b>					
0x61XX	Salto incondizionato	N	N	N	N
0x62XX	Salta se zero (Z=1)	N	N	N	N
0x63XX	Salta se non zero (Z=0)	N	N	N	N
<b>Gruppo di input/output</b>					
0x71P0	Segnala all'unità periferica di caricare il registro di input	N	N	N	N
0x72R0	Copia il byte del registro di input al registro R	D	D	N	N
0x73P0	Segnala all'unità periferica di scaricare il registro di output	N	N	N	N
0x74R0	Copia il byte dal registro R nel registro di output	D	D	N	N
<b>Operazioni miscellanee</b>					
0x8400	Nessuna operazione	N	N	N	N
0x8500	Alt	N	N	N	N

## Moduli logici:

- CPU – La CPU dell'acm-s è tutta contenuta nel file `run.c`. La funzione `run()` è il classico ciclo di processore, si compone infatti di una `fetch_word()`, di una `decode()` e di una `execute()`.
- ALU – L'unità aritmetico-logica è implementata interamente nella funzione `execute()`, che esegue tutte le operazioni matematiche e logiche ed imposta i codici di condizione.
- RAM – La memoria dell'elaboratore è semplicemente un array di caratteri (quindi ogni elemento è lungo 8 bit) di grandezza `MAXMEM` definito esternamente al `main()`.
- I/O – L'acm-s ha un sistema molto primitivo di interazione con l'ambiente esterno, basato su due registri, uno di input ed uno di output, entrambi lunghi 8bit (char) e definiti nel file `run.c`.

## Funzionamento del programma

Questo è l'albero di attivazione delle procedure all'interno del programma.



Come si può vedere, ogni voce del menù è realizzata mediante un'opportuna chiamata ad una sotto funzione.

## Organizzazione in file:

- `acm-s.h`

E' un header generico incluso in tutti i file di codice del programma.

- `main.c`

Contiene la funzione `main()`, la quale si occupa di stampare il menù e di associarne le voci alle rispettive funzioni. Gestisce anche i codici di ritorno di quest'ultime.

Inoltre definisce esternamente l'array che implementa la RAM dell'elaboratore, in modo che ogni altra funzione possa accederne se ne ha bisogno.

- `funcs.c` e `funcs.h`

Questi file contengono definizione e dichiarazione di alcune funzioni generiche che il programma necessita. Molto importanti sono la `make_word()`, la `get_hi_byte()` e la `get_low_byte()` che permettono di passare dati tra i registri a 16 bit e gli elementi dell'array della memoria, che sono invece ad 8 bit. La `wipe_mem()` semplicemente azzerava tutto l'array della RAM, la `need_operand()` è un modo per centralizzare il controllo delle istruzioni che necessitano o no di un operando aggiuntivo per poter essere eseguite, la `getbit()` ritorna il valore di un bit in una particolare posizione dell'argomento della funzione e la `disasm()` è in grado di ricavare il mnemonico di un'istruzione a partire dal codice macchina di quest'ultima.

- `load.c` e `load.h`

Azzerava la memoria con la funzione `wipe_mem()` e resetta la macchina con `reset()`.

Quindi chiama la funzione `loadfile()`, la quale, adoperando le funzioni standard di libreria, apre il file passato per argomento, ne legge il contenuto, e lo copia nella memoria dell'acm-s.

Nel fare questo si serve della `need_operand()` per sapere se l'istruzione che sta caricando necessita di un operando (che viene scritto nel file di testo del programma sulla stessa linea dell'istruzione).

- `dump.c` e `dump.h`

E' qui definita la funzione `dump()` che stampa il contenuto della memoria entro gli indirizzi indicati come argomenti. Si avvale della `need_operand()`, per stampare sulla stessa linea due parole nel caso che la seconda sia l'operando della prima, e della `disasm()` per stampare il mnemonico dell'istruzione.

- `run.c` e `run.h`

Sono qui definite tutte le funzioni che implementano realmente il funzionamento della cpu simulata. La funzione principale è la `run()`, che esegue tutte le istruzioni del programma finché non incontra quella di `Alt`. Questa funzione accetta come parametro un intero che funziona da flag per lo stato di tracing, se è impostata ad 1 la funzione esegue una sola istruzione, stampa lo stato dei registri con la `show_state()` e torna al menù.

La `run()` si compone a sua volta di altre sotto funzioni, tra cui la `fetch_word()`, che ritorna una parola di macchina da 16 bit e incrementa il PC. Vi sono poi la `decode()`, che con una serie di cicli separa le cifre esadecimali che compongono il registro di istruzione in modo che sia poi più agevole per la `execute()` eseguire le istruzioni in base a controlli sulle cifre che le compongono.

Quest'ultima adopera diverse funzioni ausiliarie come la `check_z()`, la `check_n()`, la `check_c()`, la `check_v()` e la `check_addr()` per controllare i codici di condizione e la validità degli indirizzi.

Ulteriori informazioni (come ad esempio il valore di ritorno delle funzioni) sono contenute nei commenti alle singole funzioni, all'interno dei rispettivi file sorgente.

## To-Do

C'è un grande margine di miglioramento per questo programma, ecco alcune idee del suo autore:

- Una cosa relativamente semplice, ma utile, è quella di permettere al modulo che carica i file il riconoscimento di righe vuote e di commento.
- La miglioria principale è sicuramente il poter gestire l'accesso alla memoria per singolo byte, nell'acm non semplificato, infatti, ogni istruzione che accede alla memoria ha un'istruzione corrispondente che accede al singolo byte.
- Altra cosa importante è il completamento del set di istruzioni al fine di renderlo pienamente compatibile con quello dell'acm, compresi i registri ed i modi di indirizzamento mancanti.
- Utile aggiunta potrebbe essere quella di un assembler, con conseguente riscrittura della `loadfile()` per permettere la lettura di file compilati (binari).
- Una cosa interessante da realizzare sarebbe quella di definire una variabile `char` ed usarne i singoli bit come codici di condizione (come avviene nei processori reali). Controllare ed impostare tali bit diverrebbe solo una questione di bitmask ed operazioni logiche.

