

Architettura degli Elaboratori

Come implementare il computer

HACK

Andrea Malvezzi

05 novembre, 2024

Contents

| | | |
|----------|---|----------|
| 1 | Premesse | 3 |
| 2 | La memoria | 3 |
| 2.1 | La ROM | 3 |
| 2.2 | Chip memory | 3 |
| 2.2.1 | Il chip Screen | 3 |
| 2.2.2 | Esempio di codice per disegnare un rettangolo largo 16 pixel e alto MEM[0] pixel | 4 |
| 2.2.3 | Il chip Keyboard | 5 |
| 3 | Comportamento della CPU | 5 |
| 4 | Codifica A-Instruction | 6 |

1 Premesse

Il computer HACK sfrutta l'architettura Von Neumann a 16-bit, con una feature di quella Harvard: la memoria dati viene separata da quella programma, in modo da poter caricare contemporaneamente dati e istruzioni.

Questo viene reso possibile dal bus dati che, mentre nelle architetture moderne svolge due compiti distinti e quindi non eseguibili in parallelo, qui ne svolge solamente uno. Conseguentemente per eseguire un'istruzione basta un ciclo di clock.

2 La memoria

2.1 La ROM

La ROM corrisponde ad un chip built-in che dato un address 15-bit codifica in binario un'istruzione da eseguire, quindi: $\text{out} = \text{ROM32K}[\text{address}]$.

2.2 Chip memory

Il chip memory è composto da un totale di 3 componenti, per un totale di 24K indirizzi:

- la RAM da 16K;
- il chip **Screen** da 8K, usato per mappare lo schermo;
- un singolo registro **Keyboard** per leggere il tasto premuto;

Tuttavia, avendo in ingresso un address da 15 bit da convertire in binario, si spreca molti indirizzi così facendo: $2^{15} = 32767$ meno i 24576 occupati dal chip memory, per un totale di 8191 indirizzi sprecati. Cosa accade a questi?

2.2.1 Il chip Screen

All'interno del chip Screen si ha una corrispondenza diretta tra il singolo bit e un pixel sullo schermo. In base al valore di questi bit, lo schermo viene costantemente refreshato colorando di bianco (0) o di nero (1) ogni singolo pixel.

Per impostare il pixel in posizione (row, col) dello schermo ad un certo colore,

occorrerà quindi settare il bit $col\%16$ della word all'indirizzo $Screen[row*32 + col/16]$ a 1 oppure a 0.

Ma come mai questi numeri? Anzitutto, occorre specificare che nel computer HACK lo schermo è composto da 256 righe e 512 colonne.

Questo significa che in una singola riga ci saranno 512 caratteri, raggruppati in parole (che in informatica hanno spesso una grandezza di 16 bit), da cui si evince che in una singola riga si avranno 32 parole ($512/16 = 32$). Ed ecco spiegato il **row*32**.

Lo stesso ragionamento va applicato alle righe: avendo 256 righe, si avranno 16 ($256/16 = 16$) parole per colonna, da cui **col/16**.

2.2.2 Esempio di codice per disegnare un rettangolo largo 16 pixel e alto MEM[0] pixel

```
1      @0                // A=0
2      D=M              // D=RAM[0]
3
4      @INFINITE_LOOP    // Preparo il salto
5      D; JGE            // Se D>0
6
7      @counter           // Variabile counter per iterare
8      M=D              // RAM[counter]=RAM[0]
9
10     @SCREEN            // Predefinita dal linguaggio
11     D=A               // D=Indirizzo Screen
12     @address           // Variabile address
13     M=D              // RAM[address]=Indirizzo Screen
14
15     (LOOP)
16         @address       // Variabile address
17         A=M            // A=Indirizzo Screen
18         M=-1          // MEM[Indirizzo Screen]=-1
19
20         @32            // A=32
21         D=A            // D=32
22         @address       // Variabile address
23         M=M+D          // Address+=32
24
25         @counter
```

```

26         MD=M-1          // Counter--, D per check sul jump
27
28         @LOOP
29         D; JGT           // Torno all'inizio
30
31     (END)                // Ciclo dummy finale
32         @END
33         0; JMP

```

2.2.3 Il chip Keyboard

Come annunciato precedentemente questo chip è composto da un solo registro e fornisce in output la codifica ASCII estesa del tasto premuto, oppure 0 se non si preme alcun tasto.

Il registro è read-only e vi sono alcuni tasti con codifiche particolari, come lo spazio.

3 Comportamento della CPU

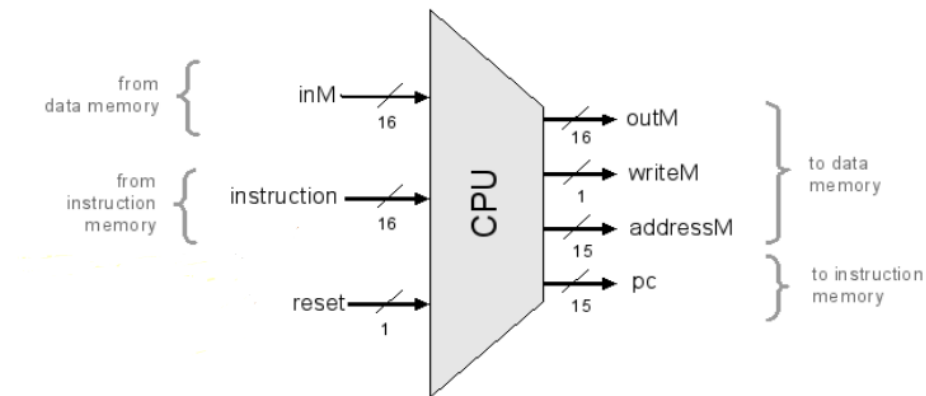


Figure 1: Schema logico CPU nell'architettura HACK.

La CPU è composta da 4 componenti: l'ALU e 3 registri, quali A, D, PC. Questa esegue istruzioni seguendo le specifiche del linguaggio HACK (nel nostro caso). Ovvero:

- i valori D ed A, se presenti, sono letti e/o scritti nei registri A ed M;
- il valore M, se presente nella right side dell'istruzione da eseguire, viene LETTO da inM (input-M);
- se invece il valore M risulta presente nella left side dell'istruzione da eseguire, allora l'output verrà scritto in outM, A prenderà il valore di addressM e writeM verrà alzato ad 1.

4 Codifica A-Instruction