Architettura con prefetching: Mic-2

Aniello Minutolo

Liberare la ALU da alcuni compiti

Sommario

1. Liberare la ALU da alcuni compiti
2. Unità di prelievo dell’istruzione
3. Architettura con prefetching: Mic-2

Esempio di ottimizzazione del microcodice

È possibile formulare in questo modo la prima tecnica che ci permette di ridurre la lunghezza del percorso: unire il ciclo d'interpretazione alla fine delle sequenze di microcodice.

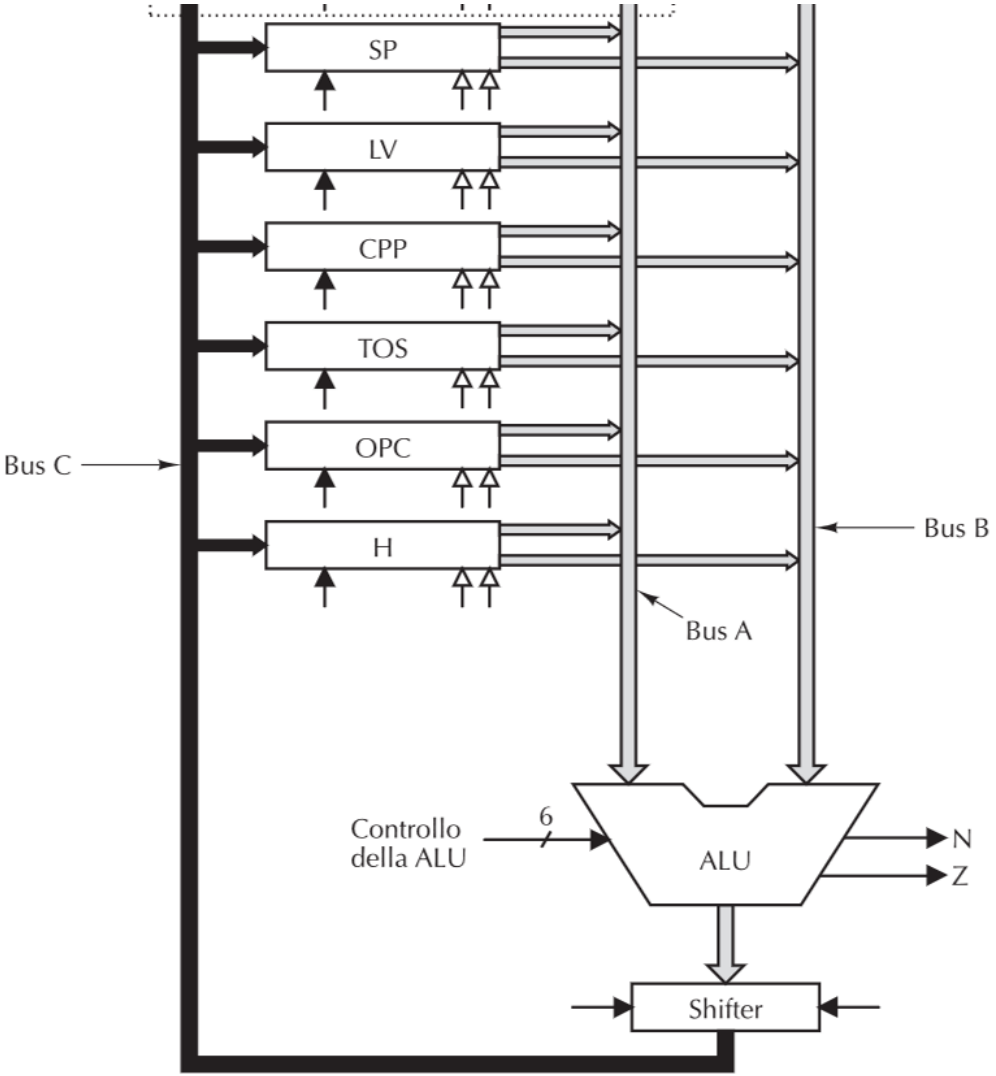
Etichetta	Operazioni	Commenti
pop1	MAR = SP = SP – 1; rd	Legge la parola sotto la cima dello stack
pop2		Attende che il nuovo TOS sia letto dalla memoria
pop3	TOS = MDR; goto Main1	Copia la nuova parola in TOS
Main1	PC = PC + 1; fetch; goto (MBR)	MBR contiene il codice operativo; prelievo del byte successivo; diramazione

Etichetta	Operazioni	Commenti
pop1	MAR = SP = SP – 1; rd	Legge la parola sotto la cima dello stack
Main1.pop	PC = PC + 1; fetch	MBR contiene il codice operativo; prelievo del byte successivo
pop3	TOS = MDR; goto (MBR)	Copia la nuova parola in TOS; diramazione a seconda del codice operativo

Architettura a 3 bus

Un’altra semplice modifica per ridurre la lunghezza del percorso di esecuzione consiste nell’utilizzo di due bus di input, A e B, per la ALU

- permette di risparmiare un ciclo in tutte quelle istruzioni che utilizzano H come registro "temporaneo"



Analizziamo le parti comuni a tutte le istruzioni

Entrambe le tecniche precedenti sono valide, ma per ottenere un netto miglioramento delle prestazioni abbiamo bisogno di qualcosa di molto più radicale

Facciamo un passo indietro e analizziamo le parti comuni a tutte le istruzioni, ovvero **il prelievo e la decodifica dei campi**

Analizziamo le parti comuni a tutte le istruzioni

Per ogni istruzione si possono verificare le seguenti operazioni:

1. il valore di PC viene fatto passare attraverso la ALU per incrementarlo
2. il valore di PC è utilizzato per prelevare il byte successivo nel flusso delle istruzioni
3. gli operandi sono letti dalla memoria
4. gli operandi sono scritti in memoria
5. la ALU esegue una computazione e i risultati vengono memorizzati

Analizziamo le parti comuni a tutte le istruzioni

Se un'istruzione contiene campi aggiuntivi (per gli operandi) questi vanno prelevati esplicitamente, un byte alla volta, e poi assemblati prima di poterli utilizzare

- prelevare e assemblare un operando impegna la ALU per almeno un ciclo per ogni byte, per incrementare PC e in seguito assemblare l'indice, o lo spiazzamento, risultante

Durante praticamente ogni ciclo la ALU viene utilizzata, oltre che per il "lavoro" vero e proprio dell'istruzione, anche per una grande varietà di operazioni relative al prelievo dell'istruzione e l'assemblaggio dei campi al suo interno

- per poter sovrapporre il ciclo principale è necessario liberare la ALU da alcuni di questi compiti

Analizziamo le parti comuni a tutte le istruzioni

In molti casi la ALU viene semplicemente usata come un cammino per copiare un valore da un registro a un altro;

- questi cicli potrebbero essere eliminati introducendo percorsi dati aggiuntivi che non passino attraverso la ALU.

Ad esempio

- potrebbe essere vantaggioso creare un percorso da TOS a MDR, oppure da MDR a TOS, dato che la parola in cima allo stack viene scambiata frequentemente tra questi due registri.

Unità di prelievo dell’istruzione

Unità di prelievo dell’istruzione

In Mic-1 è possibile eliminare gran parte del carico di lavoro che grava sulla ALU creando un’unità indipendente che si occupa del prelievo e dell’elaborazione delle istruzioni

Questa unità, chiamata **IFU** (Instruction Fetch Unit, può in modo indipendente dalla ALU

- **prelevare l’istruzione corrente** e **incrementare il PC;**
- **prelevare gli operandi (1 o 2 byte)** prima ancora di decodificare l’istruzione corrente e scoprire se ne abbia.

Unità di prelievo dell’istruzione

Una **IFU** può assemblare gli operandi a 8 e 16 bit, in modo che siano immediatamente pronti in qualsiasi momento vengano richiesti

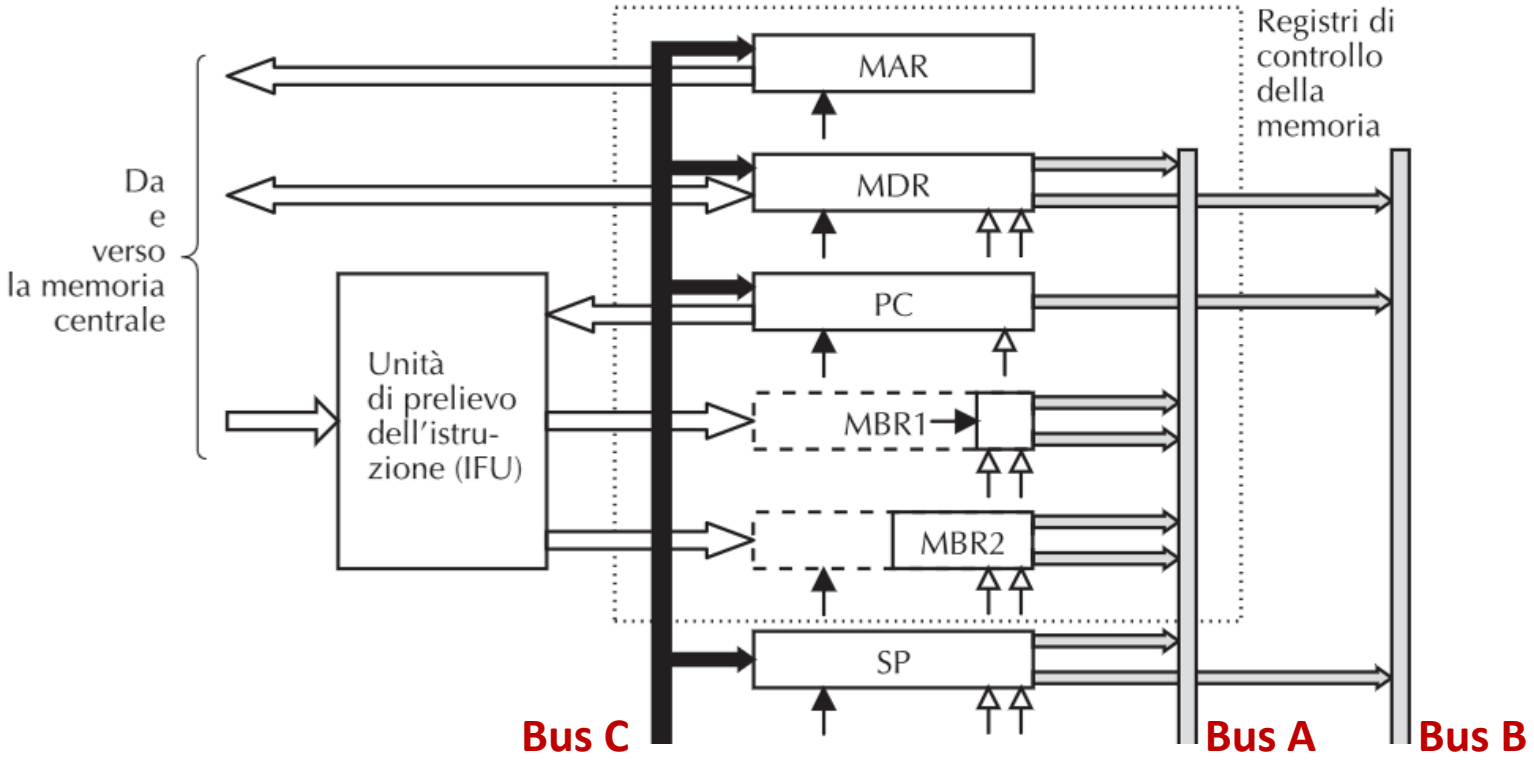
Per poter far ciò esistono almeno due modi distinti

1. la IFU può interpretare realmente ciascun codice operativo, determinando quanti campi aggiuntivi devono essere prelevati, e assemblarli in un registro;
2. la IFU legge sempre e comunque 2 extra byte (il massimo dei possibili operandi delle istruzioni) e li salva in due registri delle istruzioni MBR1 (8 bit) e MBR2 (16 bit).

Unità di prelievo dell'istruzione

Aggiungiamo una IFU nell'architettura Mic-1 a 3 bus

- al posto di un solo registro MBR a 8 bit, possiamo utilizzare i registri MBR1 a 8 bit e MBR2 a 16 bit per leggere sempre 2 extra byte
- all'inizio di ogni istruzione, MBR1 contiene il codice operativo dell'istruzione stessa



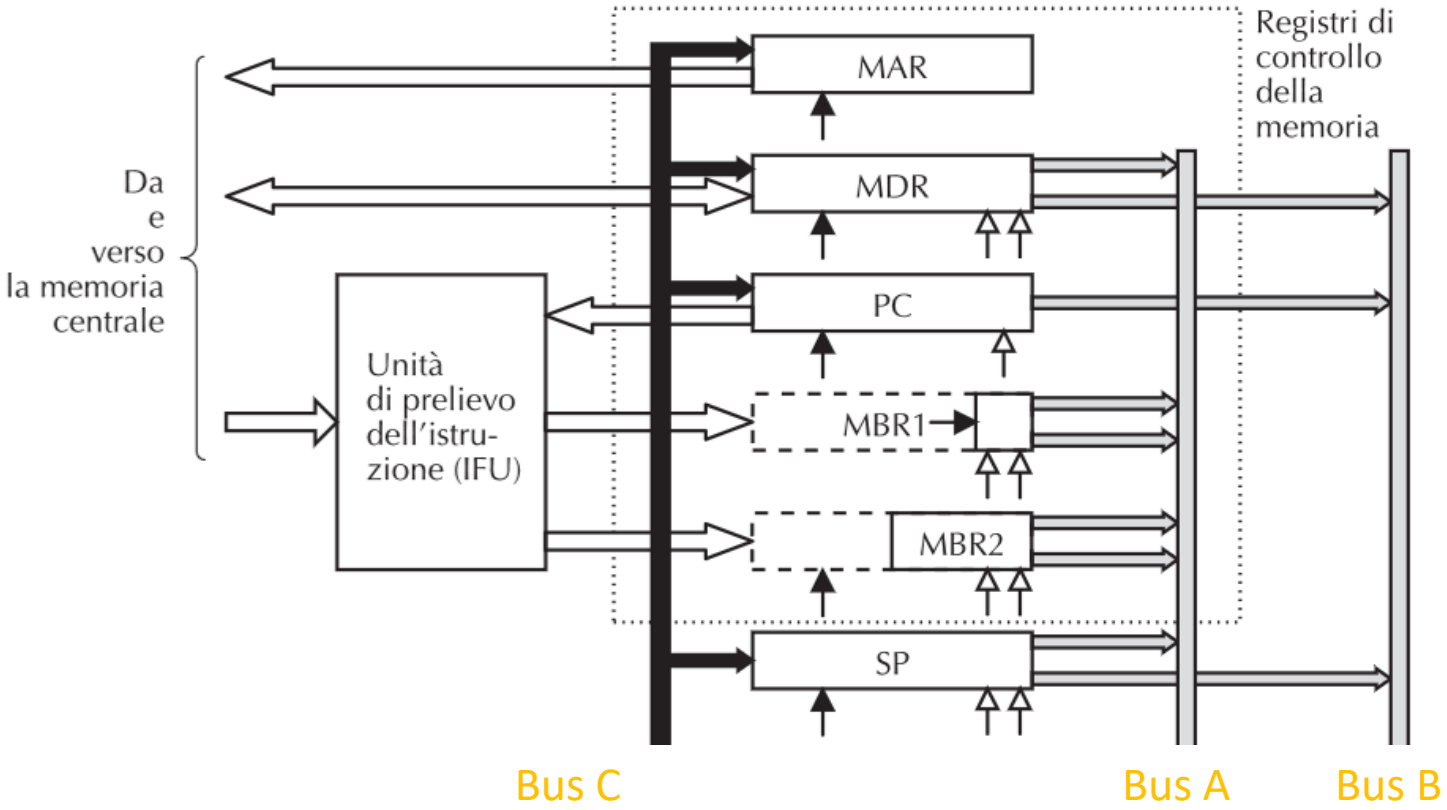
Unità di prelievo dell’istruzione

La IFU, parallelamente al normale funzionamento del data path, effettua operazioni di “pre-fetching” per caricare più istruzioni insieme

- realizza una coda di istruzioni salvate nei suoi registri, in modo che è sempre presente un’istruzione successiva da eseguire appena viene terminata l’esecuzione della precedente

Unità di prelievo dell'istruzione

La IFU possiede un proprio registro per l'indirizzo di memoria, chiamato IMAR, utilizzato per puntare all'indirizzo di memoria in cui si trova la parola da prelevare questo registro possiede un proprio incrementatore (conta le parole) e non occorre quindi utilizzare la ALU per aggiornarlo (affinché si riferisca alla parola successiva).



Unità di prelievo dell’istruzione

La IFU deve monitorare il bus C per rilevare quando il PC viene caricato e copiare di conseguenza il nuovo valore in IMAR

Utilizzando la IFU, il PC viene modificato dalla ALU solamente nel caso in cui sia necessario cambiare la natura sequenziale del flusso di byte delle istruzioni

- ciò si verifica quando viene soddisfatta la condizione di una diramazione e nel caso delle istruzioni INVOKEVIRTUAL e IRETURN.

Unità di prelievo dell’istruzione

È compito della IFU tenere aggiornato il PC, dato che il microprogramma non lo incrementa più in modo esplicito quando vengono prelevati i codici operativi

- per sapere quando occorre incrementare il PC, la IFU rileva quando viene consumato un byte del flusso dell’istruzione, cioè quando viene letto MBR1, MBR2, o le loro versioni senza segno;
- a seconda di quanti byte sono stati consumati un incrementatore aumenta il valore del PC di 1 oppure di 2, in modo che PC contiene sempre l’indirizzo del primo byte non ancora consumato.

Unità di prelievo dell’istruzione

Il fatto di non dover incrementare PC nel ciclo principale rappresenta un grande vantaggio

- spesso infatti la microistruzione nella quale viene incrementato il PC compie poco lavoro aggiuntivo;
- se si eliminano queste istruzioni si riduce quindi il percorso di esecuzione.

E’ possibile quindi ridurre la lunghezza del percorso e ottenere una macchina più veloce aumentando il numero di componenti hardware

- **una IFU dedicata al prelievo delle istruzioni dalla memoria.**

Architettura con prefetching: Mic-2

Architettura con prefetching: Mic-2

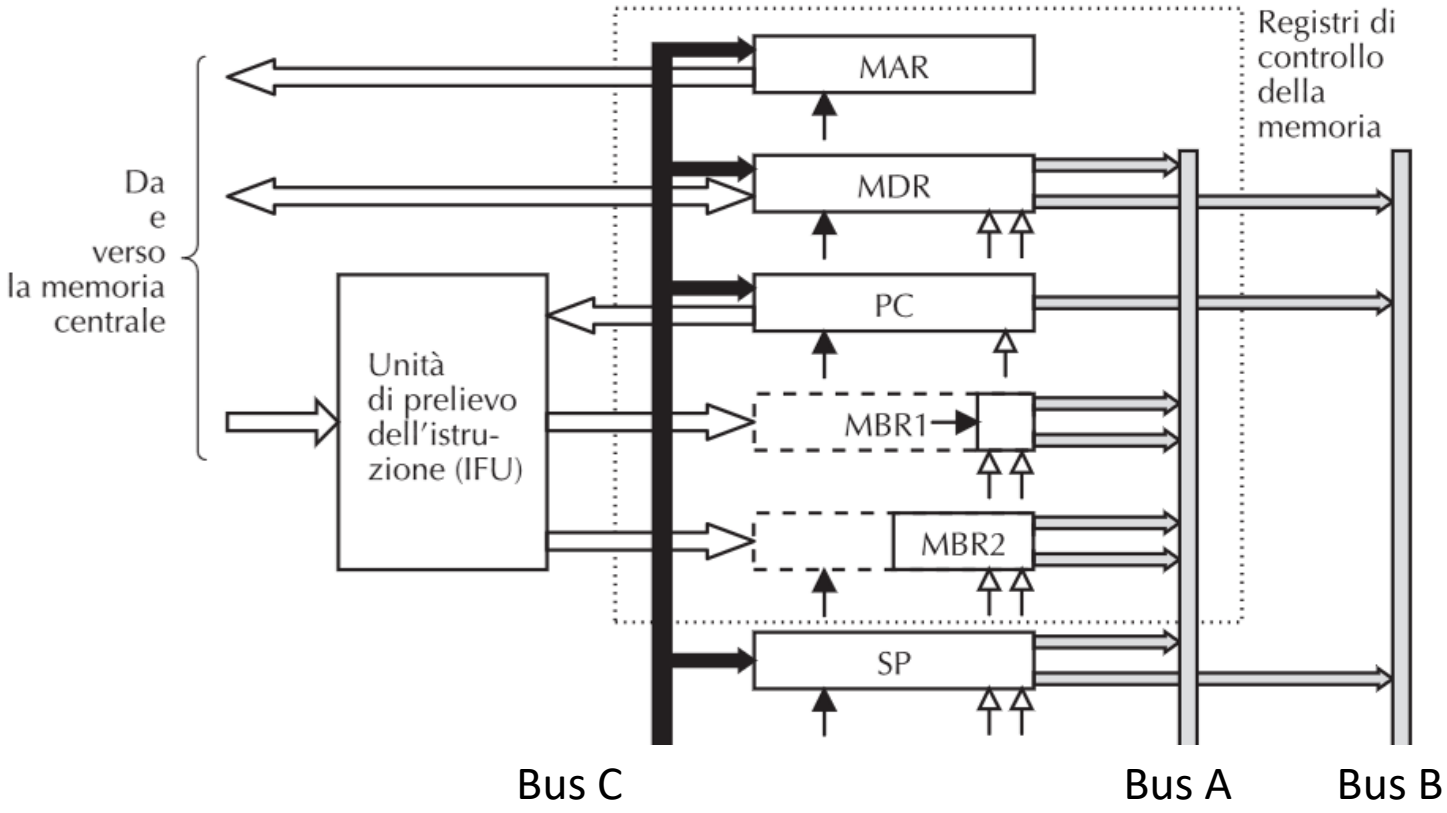
La IFU permette di ridurre in modo considerevole la lunghezza media del percorso di un'istruzione

- elimina interamente il ciclo principale, dato che alla fine di ogni istruzione si effettua un semplice salto all'istruzione successiva;
- risparmia l'utilizzo della ALU per l'incremento del PC, e riduce la lunghezza del percorso ogni volta che viene calcolato un indice o uno spiazzamento a 16 bit;
  - dato che assembla il valore a 16 bit e lo fornisce direttamente alla ALU come un valore a 32 bit, evitando di doverlo assemblare all'interno di H.

Architettura con prefetching: Mic-2

Mic-2 è una versione migliorata di Mic-1 in cui è stata aggiunta la IFU

- Il resto del data path rimane invariato



Funzionamento di Mic-2

Per vedere un esempio del funzionamento consideriamo **IADD** che sostituisce le due parole in cima allo stack con la loro somma

Microcodice di **IADD** per **Mic-1**

- durante iadd3, preleva la seconda parola dello stack, esegue l'addizione e, una volta terminata l'operazione, torna a *Main1* per incrementare PC e passare alla microistruzione successiva

Etichetta	Microistruzione	Commenti
Iadd1 iadd2 iadd3	MAR = SP = SP – 1; rd H = TOS MDR = TOS = MDR + H; wr; goto Main1	Legge la seconda parola in cima allo stack H = cima dello stack Somma le due parole in cima allo stack; scrive in cima allo stack; torna a Main1 per incrementare PC e passare alla microistruzione successiva

Funzionamento di Mic-2

Microcodice di **IADD** per **Mic-2**

- durante iadd3, preleva la seconda parola dello stack, esegue l'addizione e, una volta terminata l'operazione, non è più necessario tornare a Main1 per incrementare PC e passare alla microistruzione successiva

Quando la IFU vede che durante iadd3 è stato effettuato un riferimento a MBR1, il suo registro a scorrimento trasla tutto il proprio contenuto a destra e ricarica sia MBR1 sia MBR2

- l'unità effettua quindi una transizione verso uno stato che è inferiore di uno rispetto a quello corrente e, se il nuovo stato è 2, la IFU inizia a prelevare una nuova parola dalla memoria

Etichetta	Microistruzione	Commenti
Iadd1 iadd2 iadd3	MAR = SP = SP – 1; rd H = TOS MDR = TOS = MDR + H; wr; goto MBR1	Legge la seconda parola in cima allo stack H = cima dello stack Somma le due parole in cima allo stack; scrive in cima allo stack; torna a MBR1

Miglioramenti di Mic-2

Tutto ciò viene effettuato dall'hardware, senza alcun intervento del microcodice

- questo permette di ridurre IADD da quattro a tre microistruzioni

Mic-2 permette di ottenere miglioramenti più marcati per alcune istruzioni piuttosto che altre

- per migliorare le prestazioni globali del sistema conta migliorare le istruzioni usate più frequentemente
  - ILOAD passa da 6 microistruzioni a 3
  - IADD passa da 4 microistruzioni a 3
  - IF_ICMPEQ passa da 13 microistruzioni a 10 quando la condizione è verificata; da 10 microistruzioni a 8 altrimenti

Bibliografia

Libro di testo

- Andrew S. Tanenbaum e Todd Austin. Architettura dei calcolatori. Un approccio strutturale. 6/ED. Anno 2013. Pearson Italia spa.
  (Disponibile nella sezione “Biblioteca”)

Fonte argomenti e immagini

- Capitolo 4, Paragrafi da 4.4.2 a 4.4.3, pp. 297-304