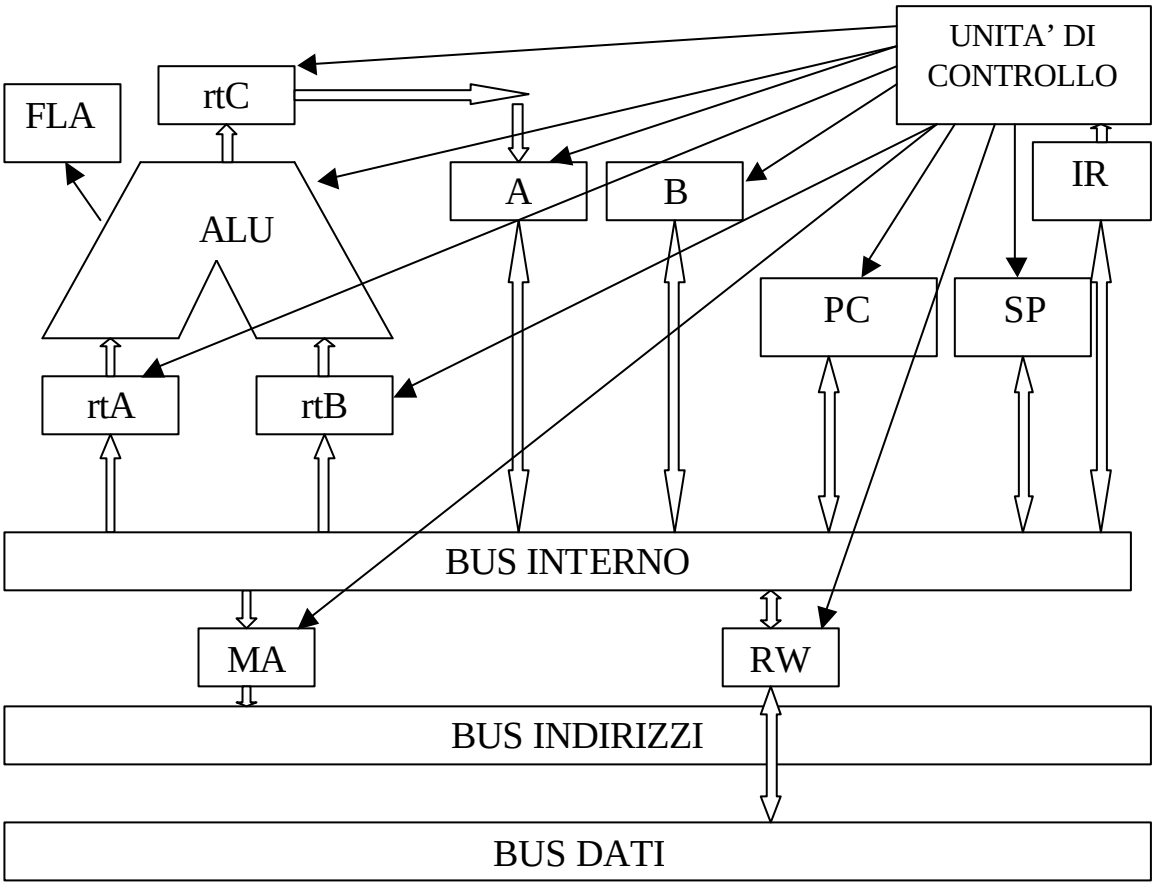
Architetture microprogrammate.

Necessità di un’architettura microprogrammata

Un’architettura microprogrammata si trova nei microprocessori di tipo CISC, cioè nei microprocessori la cui filosofia di progettazione si pone l’obiettivo di arricchirne il set di istruzione. Facciamo riferimento, per proseguire il discorso, al seguente schema di principio di un microprocessore



In un microprocessore il cervello è l’unità di controllo. Cosa succede quando viene eseguita un’istruzione? Succede che l’unità di controllo emette una serie di segnali elettrici per costringere i vari componenti presenti all’interno del microprocessore a cooperare per il compimento dell’istruzione.

Supponiamo, ad esempio, che si debba eseguire l’istruzione *SOMMA A,B*

L’esecuzione di questa istruzione consiste nei seguenti passaggi:

1. l’unità di controllo genera un segnale per costringere il registro B a scaricare il suo contenuto sul bus interno del microprocessore
2. l’unità di controllo genera un segnale di controllo per costringere il registro rtB ad acquisire il contenuto del registro B
3. l’unità di controllo genera un segnale per costringere il registro A a scaricare il suo contenuto sul bus interno del microprocessore
4. l’unità di controllo genera un segnale di controllo per costringere il registro rtA ad acquisire il contenuto del registro A
5. l’unità di controllo genera un segnale per costringere l’ALU ad eseguire l’operazione di somma
6. l’unità di controllo genera un segnale di controllo per costringere il registro rtC ad inviare il suo contenuto al registro A

Tutto questo è possibile se all’interno dell’unità di controllo esiste un circuito elettronico in grado di generare tali segnali di controllo.

Ciò significa che inserire una nuova istruzione nel set di istruzioni di un microprocessore significa complicare l’architettura dell’unità di controllo introducendo in essa un nuovo circuito elettronico in grado di eseguire tale istruzione.

Vi è allora un limite all’introduzione di nuove istruzioni nel set del microprocessore, costituito dallo spazio fisico presente nel chip.

Il problema può essere aggirato introducendo le architetture microprogrammate.

Cos'è un'architettura microprogrammata?

Una istruzione del set di istruzioni di un microprocessore, che da questo momento in poi chiamiamo *macroistruzione,* si compone di diverse fasi elementari, che sono spesso in comune con altre macroistruzioni.

Ad esempio, la macroistruzione SOMMA A, B si compone dei seguenti passaggi

- sposta il contenuto di A in rtA
- sposta il contenuto di B in rtB
- esegui la somma
- sposta il risultato in A

invece la macroistruzione SOTTRAI A, B si compone dei seguenti passaggi

- sposta il contenuto di A in rtA
- sposta il contenuto di B in rtB
- esegui la sottrazione
- sposta il risultato in A

Come si può notare, ben tre fasi elementari su quattro sono identiche per entrambe le macroistruzioni. Si ha allora l'idea di realizzare all'interno dell'unità di controllo, circuiti elettronici che realizzino soltanto queste fasi elementari e ideare un meccanismo che consenta di realizzare una macroistruzione componendo in modo opportuno queste fasi elementari.

Questo meccanismo consiste nell'introdurre all'interno dell'unità di controllo, una memoria ROM in cui saranno registrati dei programmi composti da opportune

sequenze di microistruzioni. Ogni microistruzioni costringerà l'unità di controllo a generare i segnali di controllo necessari a svolgere una fase elementare.

L'insieme di questi programmi prenderà il nome di *firmware*. L'esecuzione di una macroistruzione consisterà nell'esecuzione del corrispondente *microprogramma*.

Struttura di una microistruzione.

Con riferimento ad un microprocessore ideale come quello descritto dalla figura precedente possiamo individuare la seguente struttura per una microistruzione.

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP

Il campo ALU consiste in un certo numero di bit che codificano l'operazione che deve svolgere l'ALU. Supponendo che l'ALU sappia svolgere le operazioni di

- somma
- sottrazione
- and
- or
- xor
- incremento
- decremento
- sift a destra
- shift a sinistra

il campo ALU dovrà essere costituito da almeno 4 bit in modo da poter codificare 9 operazioni diverse. Una possibile tabella dei valori diversi che può contenere tale campo è la seguente

VALORE	*SIGNIFICATO*
0000	Nessuna operazione
0001	Somma
0010	Sottrazione
0011	And
0100	Or
0101	Xor
0110	Incremento
0111	Decremento
1000	Shift a destra
1001	Shift a sinistra
1010	Non utilizzata
1011	Non utilizzata
1100	Non utilizzata
1101	Non utilizzata
1110	Non utilizzata
1111	Non utilizzata

Il campo SRG (Sorgente) e il campo DEST (destinazione) sono dei campi utilizzati dalle microistruzioni che devono provocare il trasferimento di dati da un registro all'altro. Il campo SRG dovrà indicare da quale registro dovrà partire il dato e il campo DEST dovrà indicare in quale registro dovrà finire il dato. Poiché nel nostro microprocessore ideale (vedi figura) vi sono 11 registri, i due campi dovranno contenere almeno 4 bit. Una possibile soluzione è la seguente

VALORE	*SIGNIFICATO*
0000	Nessun registro
0001	PC
0010	SP
0011	MAR
0100	RWR
0101	IR
0110	A
0111	B
1000	FLAG
1001	rtA
1010	rtB
1011	rtC
1100	Non utilizzata
1101	Non utilizzata

1110	Non utilizzata
1111	Non utilizzata

Il campo MEM serve a generare il segnale di attivazione per la memoria quando il microprocessore vuole dialogare con essa. Basterà un solo bit per questo campo.

VALORE	*SIGNIFICATO*
0	Nessuna operazione in memoria
1	Accesso in memoria

Il campo RW, associato al campo precedente, indica, se si accede in memoria, se si tratta di un'operazione di lettura o scrittura

VALORE	*SIGNIFICATO*
0	lettura
1	scrittura

Il campo MAR serve a costringere il registro MAR (che contiene l'indirizzo della locazione di memoria con cui il microprocessore vuole dialogare) a scaricare il suo contenuto sul bus indirizzi esterno quando il microprocessore vuole dialogare con la memoria.

VALORE	*SIGNIFICATO*
0	Nessuna azione

1	Contenuto del MAR sul bus indirizzi

Il campo JUMP serve a cambiare la sequenza di esecuzione del programma effettuando un salto ad una microistruzione diversa da quella che segue in memoria. Nel nostro modello semplificato supponiamo che siano possibili solo salti incondizionati.

VALORE	*SIGNIFICATO*
00	Nessun salto
01	salto

Ma dove saltiamo? Verrà effettuato un salto all'istruzione il cui indirizzo sarà contenuto nel campo NEXT. Questo campo conterrà dunque l'indirizzo della prossima istruzione da eseguire. L'ampiezza di tale campo dipende dunque da quante locazioni sono contenute nella memoria ROM interna all'unità di controllo. Nel nostro caso supponiamo che la ROM contenga 256 locazioni per cui il campo NEXT dovrà comprendere 8 bit

Esempi di microprogrammi

Esempio 1

Vediamo come esempio il microprogramma che esegue la fase di fetch dell'istruzione. Essa si compone delle seguenti fasi elementari

- Spostare Contenuto del PC nel MAR[1]
- Spostare Contenuto del MAR sul Bus Indirizzi
- Spostare Contenuto del Bus Dati nel READ/WRITE REGISTER
- Incrementa il PC[2]

La prima microistruzione esegue un trasferimento di dati fra due registri per cui avrà il seguente valore

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00000000	0	0000	0	0	0001	0011	00

La seconda microistruzione deve soltanto costringere il registro MAR a scaricare il suo contenuto sul bus indirizzi

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00000001	1	0000	0	0	0000	0000	00

La locazione di memoria selezionata pone sul bus dati il suo contenuto, cioè il codice operativo dell'istruzione da eseguire, che deve essere letta dal microprocessore, per far ciò dobbiamo eseguire una microistruzione in cui il campo MEM sia ad 1 e il campo RW sia a 0 per effettuare una lettura

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP

[1] Infatti il PC contiene l'indirizzo della prossima istruzione da eseguire, che va trasferito nel MAR in modo da poterlo porre sul bus indirizzi

[2] ricordiamo che alla fine di ogni fase di fetch il program counter si deve incrementare automaticamente di 1 in modo da contenere l'indirizzo della istruzione da eseguire successivamente

00000010	0	0000	1	0	0000	0000	00

Ora l'istruzione è nel registro RWR e va spostata nel registro IR

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00000011	0	0000	0	0	0100	0101	00

Per incrementare il PC lo dobbiamo portare al registro di ingresso rtA dell'ALU

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00000100	0	0000	0	0	0001	1001	00

Ora comandiamo all'ALU l'operazione di incremento

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00000101	0	0111	0	0	0000	0000	00

Spostiamo il risultato da rtC a PC

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00000110	0	0000	0	0	1011	0001	00

Esempio 2

Vogliamo realizzare la macroistruzione SOMMA A,B

- Sposta A in rtA
- Sposta B in rtB
- Esegui la somma
- Sposta il risultato da rtA ad A

La prima microistruzione deve provocare lo spostamento del contenuto di A in rtA

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00000111	0	0000	0	0	0110	1001	00

La seconda microistruzione provoca lo spostamento del contenuto di B in rtB

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00001000	0	0000	0	0	0111	1010	00

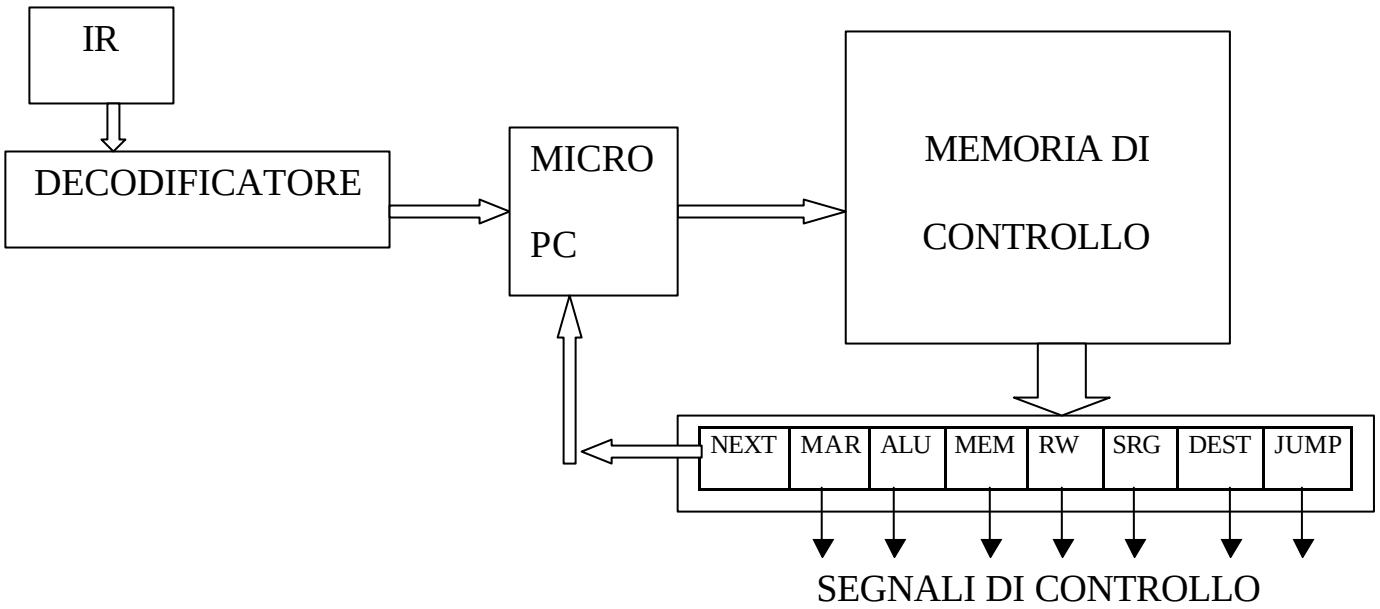
Ora comandiamo all'ALU di eseguire la somma

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00001001	1	0001	0	0	0000	0000	00

Spostiamo il risultato da rtC a A. Ora però l'esecuzione della macroistruzione è completata per cui si deve saltare alla fase di fetch che inzia con l'istruzione alla locazione 00000000

NEXT	MAR	ALU	MEM	RW	SRG	DEST	JUMP
00000000	0	0000	0	0	1011	0110	01

Struttura di un'unità di controllo microprogrammata



L’unità di decodifica , in questo caso, non deve far altro che individuare l’indirizzo della ROM da cui inizia il microprogramma che esegue la macroistruzione che si trova nel registro IR. Tale indirizzo va in un micro program counter che garantirà il prelievo delle varie microistruzioni nell’ordine corretto. La microistruzione individuata dal micro program counter viene spostata in un registro micro IR dal quale si dipartono i segnali di controllo che comandano i vari componenti del microprocessore. Se il campo JUMP è attivo, il contenuto del campo NEXT viene spostato nel micro PC, in modo da alterare la sequenza di esecuzione delle microistruzioni.