MIC-2

Il principale vantaggio del MIC-2 rispetto al MIC-1 è di ottimizzare le tempistiche di svolgimento delle operazioni.

Come si identifica il miglioramente delle istruzioni?

- Il PC è inviato alla ALU ed incrementato.
- Il PC è usato per fare il fetch del prossimo byte nel flusso che compone il programma.
- Gli operando sono letti in memoria.
- Sono scritti in memoria
- La ALU da i calcoli e il risultato viene memorizzato in memoria.

Cosa coinvolge la ALU

- Esegue un lavoro vero e proprio dell'istruzione.
- Trasgerisce i valori da un registro all'altro.
- Tratta i byte che codificano gli eventuali operandi.

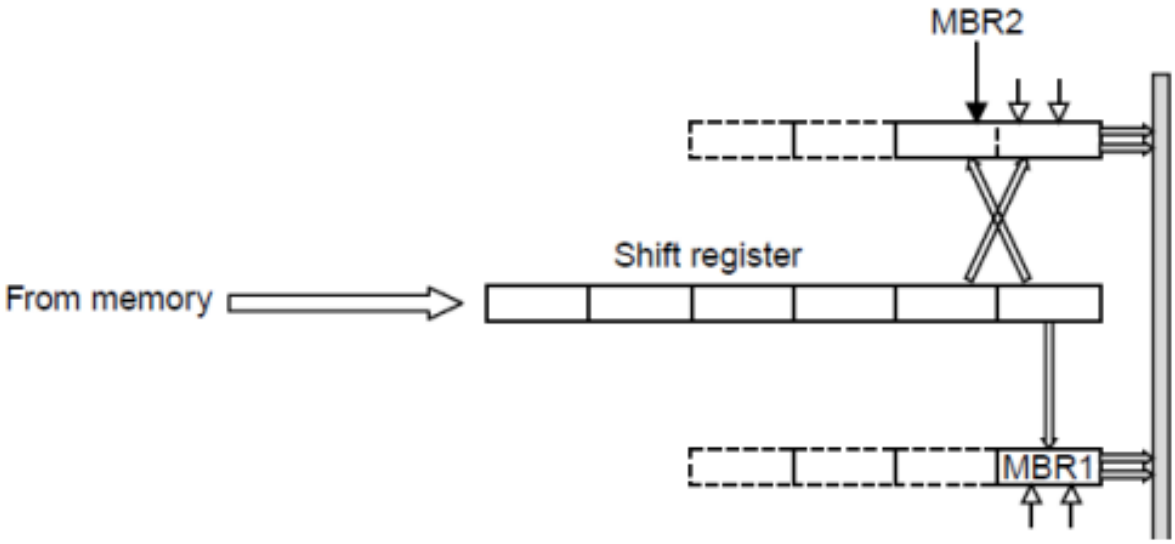
Per risolvere questo carico di lavoro troppo elevati si è pensato di aggiungere una seconda ALU la quale serve per introdurre una nuova componente chiamata "**Fetch Unit**".

Fetch Unit

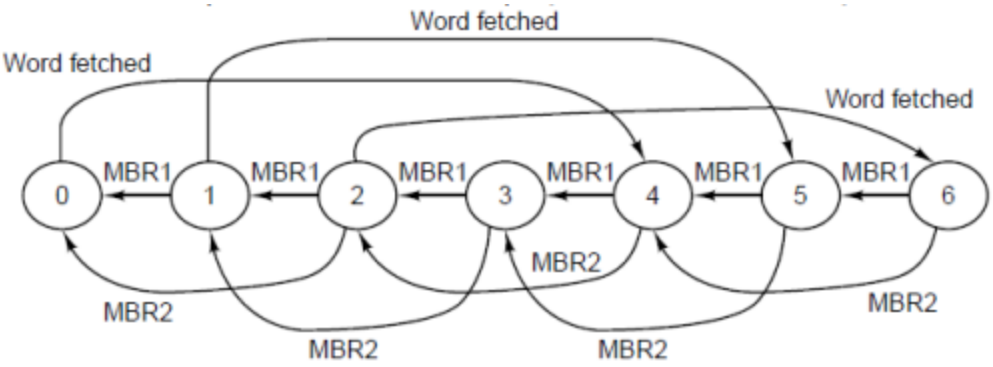
Serve per incrementare in modo indipendete del PC, interpreta ogni opcode terminando i parametri richiesti e li mette poi in un registro in modo che il microcodice abbia già a disposizione questi valori. Rendepiù facilmente disponibile i cambi da 8 o 16 bit.

Il registro a 8 bit MBR è sostituito da: MBR1 e MBR2 il primo come sempre si interfaccia con il bus B a 8 bit, il secondo è un registro a 16 bit verso il bus B.

Lo **shifter register** ha il compito di mettere il byte più significativo nel byte meno significativo in MBR2.



La macchina si può trovare in **7 stati finiti** e **3 stati di transizione**, uno alla volta. Il cambio di stato è determinato da un avvenimento che porta alla sua evoluzione. Gli eventi di transizione sono: “è stato letto il byte 1”, “è stato letto il byte 2” “voglio la parola successiva”.



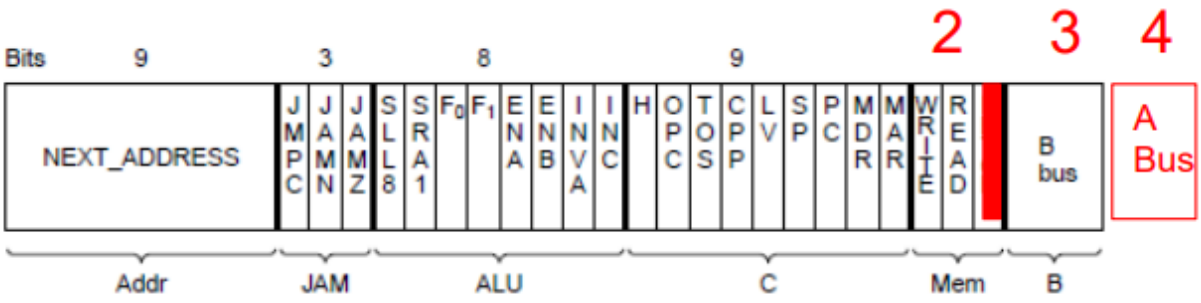
Quando il **PC viene cambiato** la macchina a stati si deve **azzerare** (svuotato).

L’**Instruction Fetch Unit** (**IFU**) ha un proprio registro per indirizzare alla parola successiva ovvero **IMAR** (**Instruction Memory Address Register**) che possiede un

contatore e quando PC è modificato dalla ALU un nuovo valore è copiato in IMAR.

L'interazione IFU e Decode & Execute funzionano in modo asincrono: quando IFU recupera un istruzione da eseguire lo inserisce nel buffer, D&E consumano 1 o 2 byte alla volta. Per la gestione del salti c'è un feedback.

IFU permette di eliminare Main1, risparmia la ALU dall'incrementare PC e riduce la lunghezza per percorso ogni volta che c'è un offset di 16 bit.



Ecco come MIC-2 estende MIC-1, riducendo il numero di pin per il bus B dato che solo 8 registri sono collegati. Sono necessari 4 bit per il bus A con 11 segnali di output enebled e bisogna introdurre un nuovo decoder per il bus A.

	MIC1	MIC2
BIPUSH byte	*4*	*2*
DUP	*3*	*2*
GOTO offset	*7*	*4*
IADD	*4*	*3*
IAND	*4*	*3*
IFEQ offset	*8(false)-11(true)*	*6(false)- 8(true)*
IFLT offset	*8(false)-11(true)*	*6(false)- 8(true)*
IF_ICMPEQ offset	*10(false)-13(true)*	*8(false)-10(true)*
IINC varnum const	*7*	*3*
ILOAD varnum	*6*	*3*
INVOKEVIRTUAL disp	*23*	*11*
IOR	*4*	*3*
IRETURN	*9*	*8*
ISTORE varnum	*7*	*5*
ISUB	*4*	*3*
LDC_W index	*8*	*3*
NOP	*2*	*1*
POP	*4*	*3*
SWAP	*7*	*6*
WIDE ILOAD	*9*	*5*
WIDE ISTORE	*10*	*7*

Ecco alcune migliorie riportate da MIC-2 rispetto a MIC1 sul numero di giri necessari per finire una istruzione. Noi dobbiamo concentrarci sulla frequenza dell'uso di ogni istruzione.

Occorre quindi fare dei benchmark ovvero una serie di applicazioni svolte sia su una piattaforma che su un'altra.

Anche lo stile di programmazione può variare il risultato.