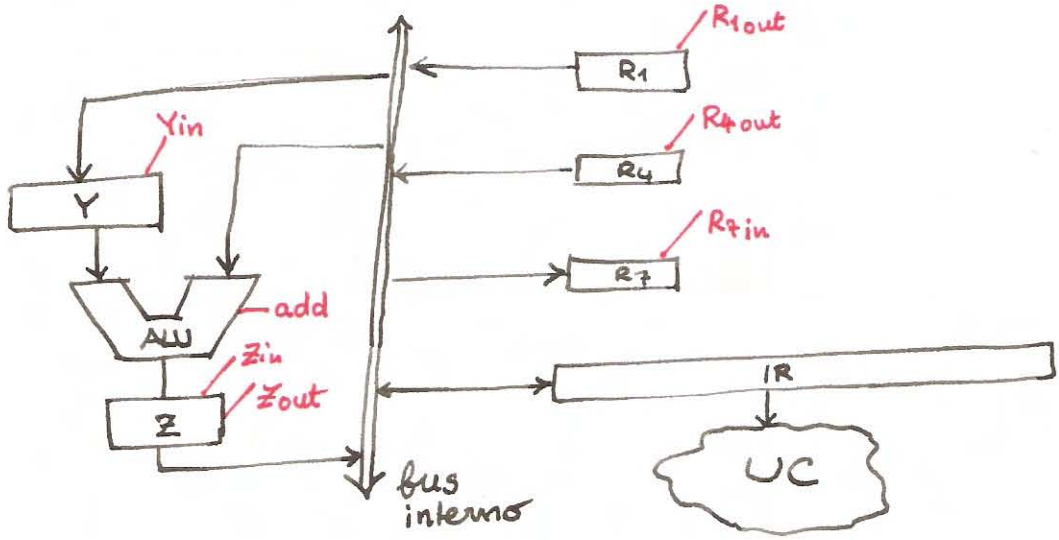
[Riassunto della lezione.] (cfr. foto, che coprono l'intera lezione)

Lez. mer. 19/11/14 (3h)



Il programmatore scrive l'istruzione assembler

ADD R_7, R_1, R_4 ; $R_7 \leftarrow R_1 + R_4$

Questo è un caso particolare dell'istruzione di macchina

ADD $R_{dest}, R_{src1}, R_{src2}$,

la cui codifica NON richiede di specificare i modi di indirizzamento degli operandi, visto che l'unico modo di indirizzamento previsto è il "registro". Quindi una codifica plausibile è

opcode (AM_{dest} AM_{src1} AM_{src2}) ID_{dest} ID_{src1} ID_{src2}

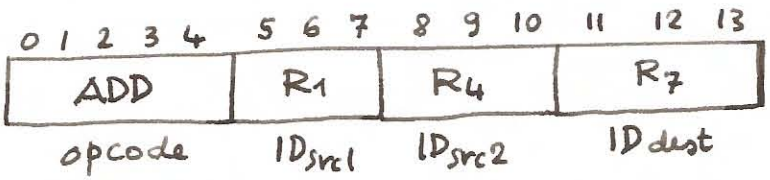
opcode → suff. 5 bit (set di istruzioni con $N \leq 2^5 = 32$ istruzioni)

(AM_{dest} AM_{src1} AM_{src2}): 0 bit per codificare questa informazione: è già nota in anticipo!

ID_{dest} ID_{src1} ID_{src2}: 3 3 3

3 bit per ciascun registro, ipotizzando che i registri di uso generale siano 8: $R_0 \ldots R_7$

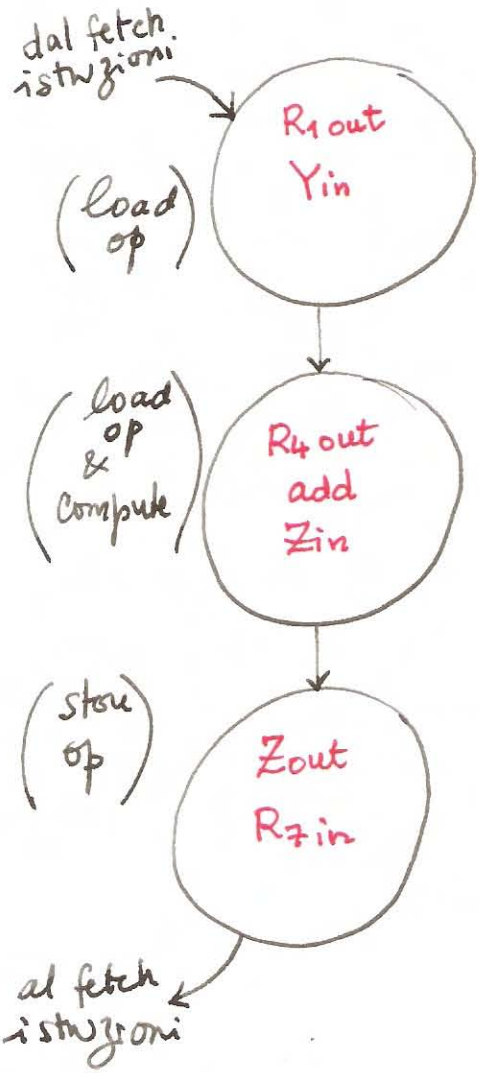
Utilizziamo una codifica alternativa (ci verrà utile dopo), scambiando le ID di sorgente e destinazione:



lunghezza: 14 bit

N.B. L'ordine di codifica è indipendente dall'ordine di sintassi dell'istruzione: quella sopra è comunque la codifica dell'istruzione ADD R_7, R_1, R_4, che segue la convenzione "destination first".

L'automa di controllo relativo all'esecuzione di questa istruzione è il seguente:



dove:

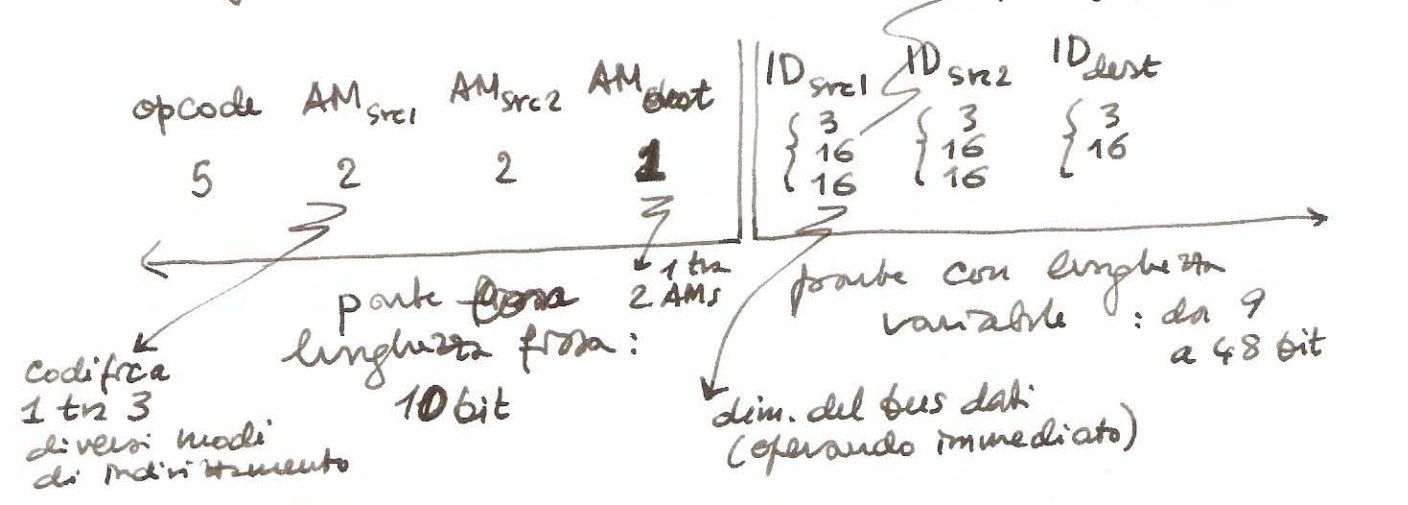
$$R_1\,out = RF\left(IR[5\text{-}7]\right)_{out}$$
$$R_4\,out = RF\left(IR[8\text{-}10]\right)_{out}$$
$$R_7\,in = RF\left(IR[11\text{-}13]\right)_{in}$$

con $RF(\bullet)$ funzione combinatoria che decodifica il registro da impiegare utilizzando i 3 bit di codifica del registro contenuti nel relativo campo dell'istruzione

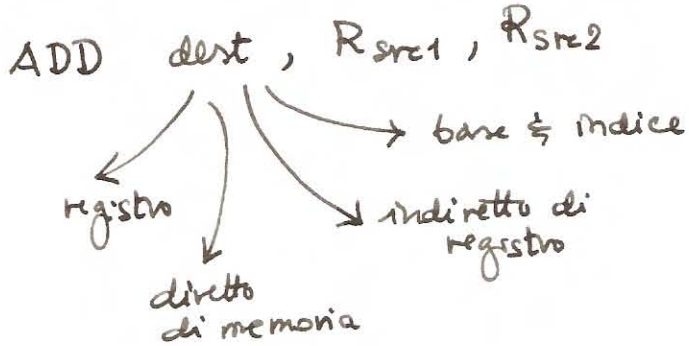
Ora: se il <u>modello di programmazione</u> (cioè l'insieme delle risorse della macchina visibili e utilizzabili dal programmatore, che include il set delle istruzioni, i registri d'uso generale, etc.) consente di eseguire la somma tra operandi qualsiasi, quindi anche non tutti registri, allora il modello dell'istruzione da codificare e da implementare nel controllo cambia:

ADD dest, src1, src2 ; dest ← src1 + src2

In questo caso, più generale del precedente, supponiamo che src1, src2 e dest possano essere specificabili con 3 diversi modi di indirizzamento: 1) registro, 2) diretto di memoria, 3) immediato. [L'unica limitazione sarà che la destinazione <u>non</u> può essere un operando immediato, naturalmente.] Quale sarà la codifica più lunga per l'istruzione, e quale la più corta? In generale, la codifica sarà



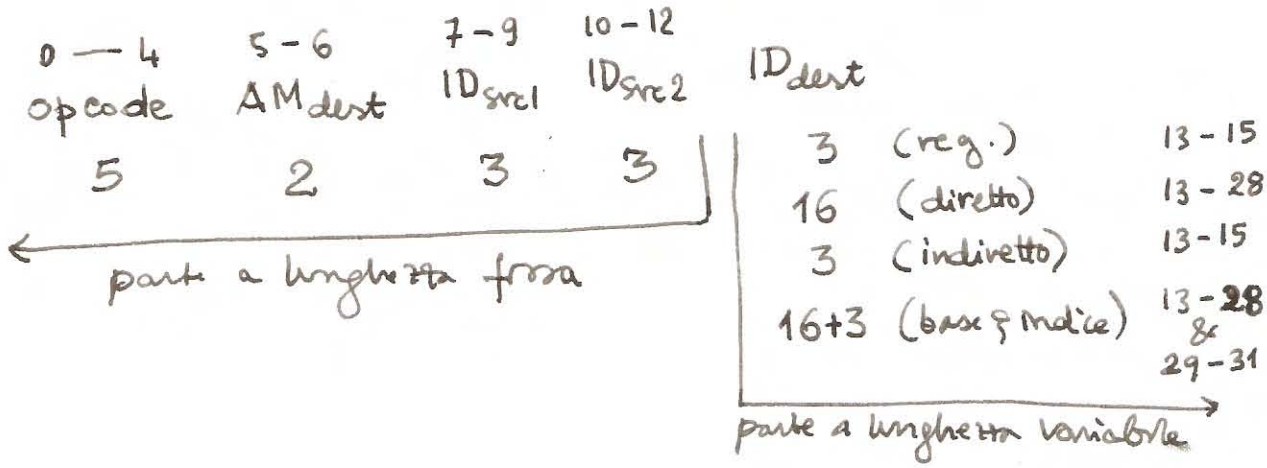
Consideriamo ora un'istruzione ADD di complessità intermedia tra la prima e la seconda viste, e progettiamola completamente (codifica + controllo):



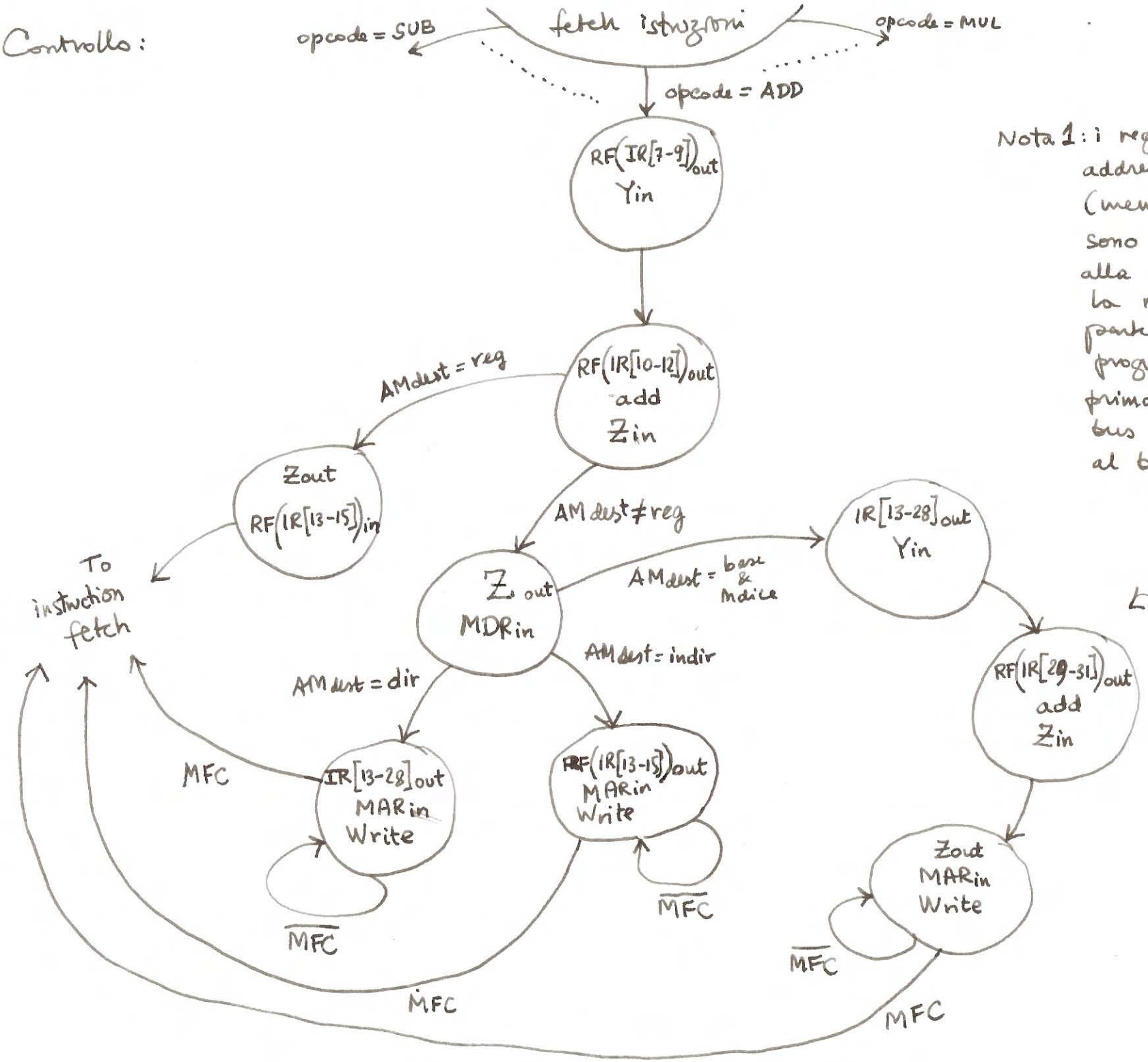
L'unico operando che può non essere un registro è quello destinazione. Esso può avere quattro diversi modi di indirizzamento:

	AM	esempio di sintassi	descrizione
	registro	R_5	il dato è in un registro di CPU
dato in memoria	diretto di memoria	Var	il dato è in memoria, nella cella all'indirizzo address(Var)
dato in memoria	indiretto di registro	(R_5)	L'indirizzo di memoria è un registro di CPU
dato in memoria	base & indice	$Vect(R_5)$	L'ind. di memoria per reperire il dato è address(Vect) + R_5

Vediamo le possibili codifiche dell'istruzione:



Controllo:



Nota 1: i registri MAR (memory address register e MDR (memory data register) sono registri di supporto alla comunicazione con la memoria, e non fanno parte del modello di programmazione. Il primo è collegato al bus indirizzi, il secondo al bus dati.

Nota 2: nell'indirizzamento base & Indice, l'indirizzo del dato non è già pronto: va calcolato. Questo allunga la durata dell'istruzione.

Confrontiamo ora i vari modi di indirizzamento per questa istruzione; non essendoci (operandi) sorgente di memoria, non ci sono cicli di bus relativi al Load operandi.

modo di indirizzamento AM_{DEST}

	registro (0)	diretto (1)	indiretto di registro (2)	base & indice (3)
Lunghezza istruzione	16	29	16	32
cicli di bus per fetch istruzione	1	2	1	2
cicli di bus per store destinazione	0	1	1	1
cicli di macchina richiesti per l'esecuzione	3 → h_0	$4+n_w \cdot 1$ → h_1	$4+n_w \cdot 1$ → h_2	$6+n_w \cdot 1$ → h_3
cicli di bus per fetch istr. + esecuzione	1	3	2	3
N_{TOT}	$3+K+n_w \cdot 1$ (1st)	$4+K \cdot 2+n_w \cdot 3$ (3rd)	$4+K \cdot 1+n_w \cdot 2$ (2nd)	$6+K \cdot 2+n_w \cdot 3$ (4th)

I cicli di macchina richiesti dal fetch istruzioni

Sono $N_{fetch} = K \cdot \#\text{cicli bus}_{fetch} + n_w \#\text{cicli bus}_{fetch}$

Quindi il totale dei cicli di macchina (fetch + esecuz.) con $AM_{dest} = i$ è

$$N_{TOT} = N_{fetch} + N_{exe} = h_i + n_w \cdot \#\text{cicli bus}_{exe} + n_w \cdot \#\text{cicli bus}_{fetch} + K \cdot \#\text{cicli bus}_{fetch} =$$

$$= h_i + K \#\text{cicli bus}_{fetch} + n_w \cdot \#\text{cicli bus}_{totali}$$