SCO-SCA A CICLI DI CLOCK MULTIPL

hp: ▸ viene eseguita una sola istruzione alla volta

▸ viene eseguito un singolo passo dell'istr per ogni ciclo di CK.

↓ CONSEGUENZA

- 1 sola ALU
- 1 sola memoria CACHE (istruz + dati)
- il codice dell'istruzione deve essere mantenuto per tutte le fasi del ciclo istruzione ad eccezione naturalmente del fetch → REGISTRO (INSTRUCTION REGISTER)

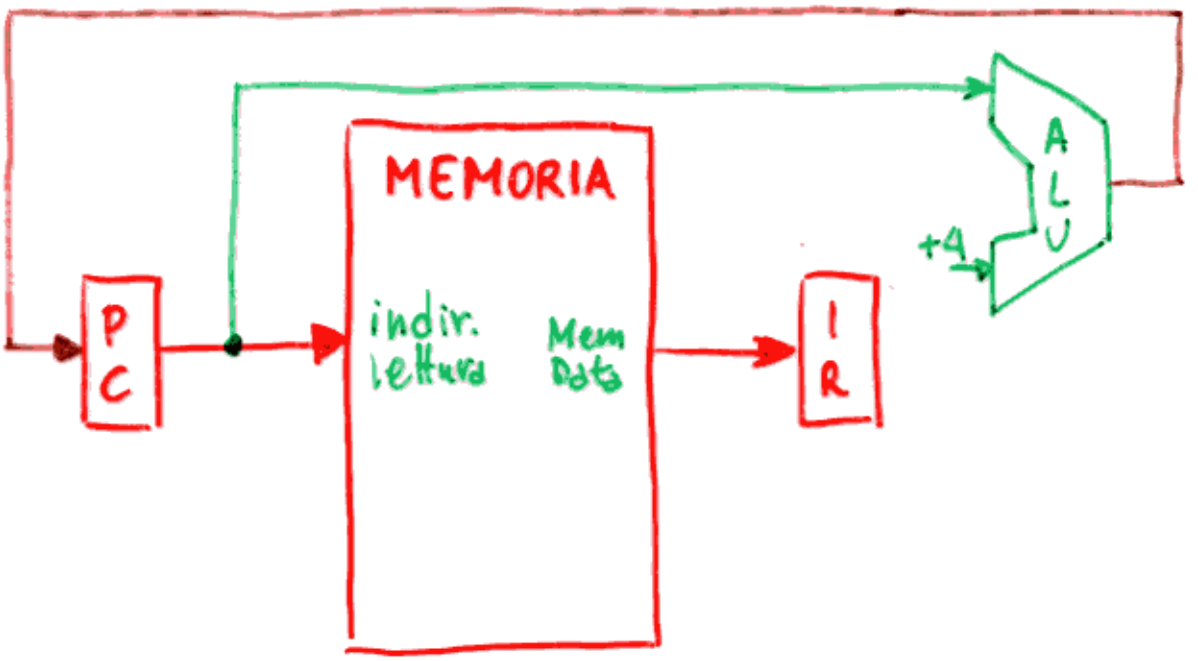
PROGETTO SCO-SCA

SCOMPOSIZIONE IN PASSI DELL'ESECUZIONE DELLE ISTRUZIONI

1° PASSO: fetch dell'istruzione

IR ← MEMORIA[PC]

PC ← PC+4



2° PASSO: preparazione esecuzione istruzione

considerazioni preliminari nel caso di una istruzione di tipo:

a) - R : bisogna leggere due registri

A = registro [IR [25-21]]
B = registro [IR [20-16]]

b) - lw/sw: bisogna leggere il registro

A = registro [IR [25-21]]

c) - branch equal: bisogna leggere due registri

a { A = registro [IR [25-21]]
B = registro [IR [20-16]]

ed eventualmente saltare all'istruzione di indirizzo

b { PC + (esteus. in segno (IR [15-0] << 2))

d) - Jump: bisogna saltare a

$(PC)_{31-28} \cdot IR[25-0] << 2$

N.B. per i salti nel caso precedente avevamo

- PC+4 + (esteus. in segno (IR [15-0] << 2))
- $(PC+4)_{31-28} \cdot$ IR[25-0] << 2

in questo caso il PC è già incrementato di 4

Le attività:

a) lettura del contenuto di due registri

b) lettura del contenuto di un registro

c)a). lettura del contenuto di due registri
• calcolo dell'eventuale indirizzo successivo

vengono effettuate nel secondo passo

Le attività:

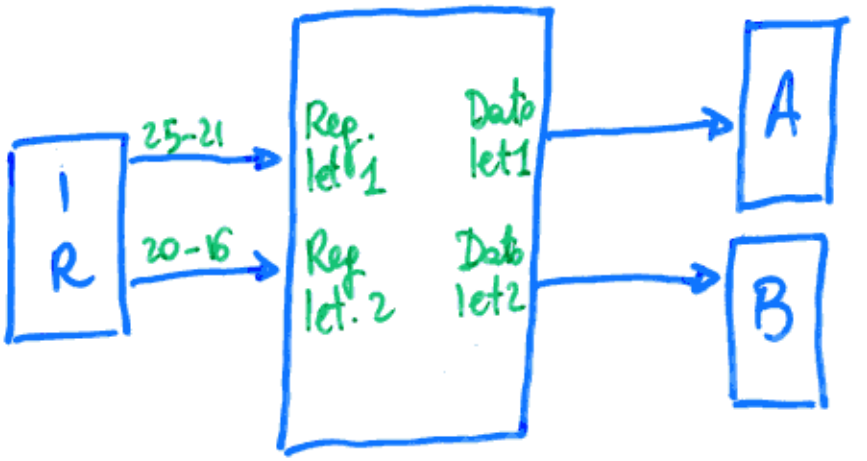
c) b) eventuale aggiornamento PC

d) aggiornamento PC

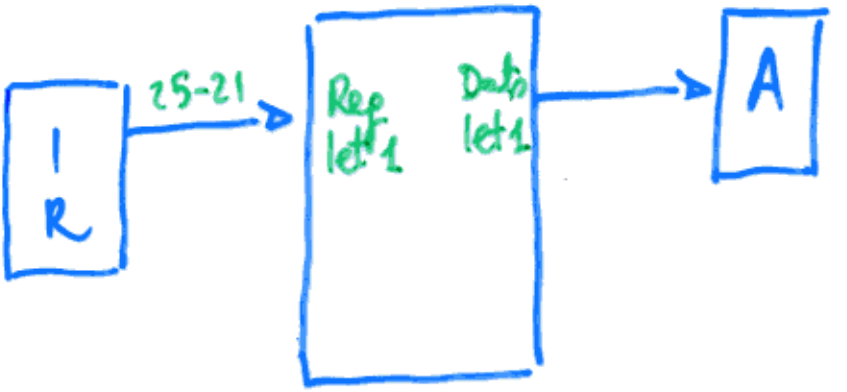
vengono effettuate nel terzo passo

2° passo

SCA x a)

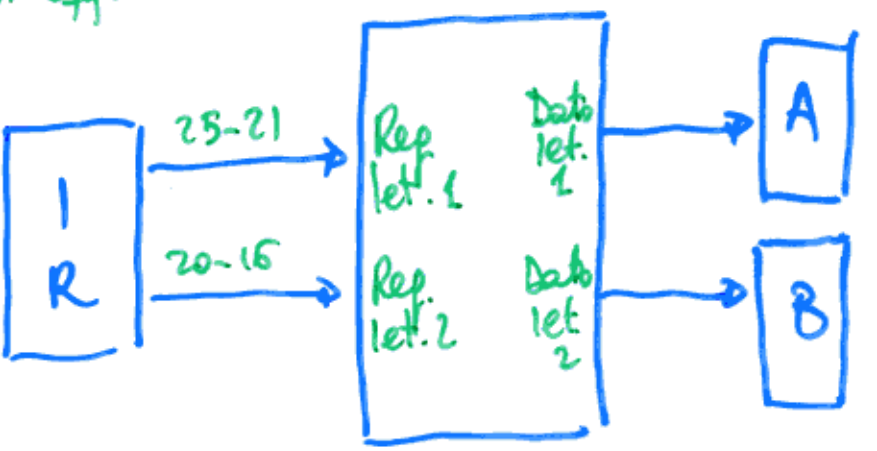


SCA x b)



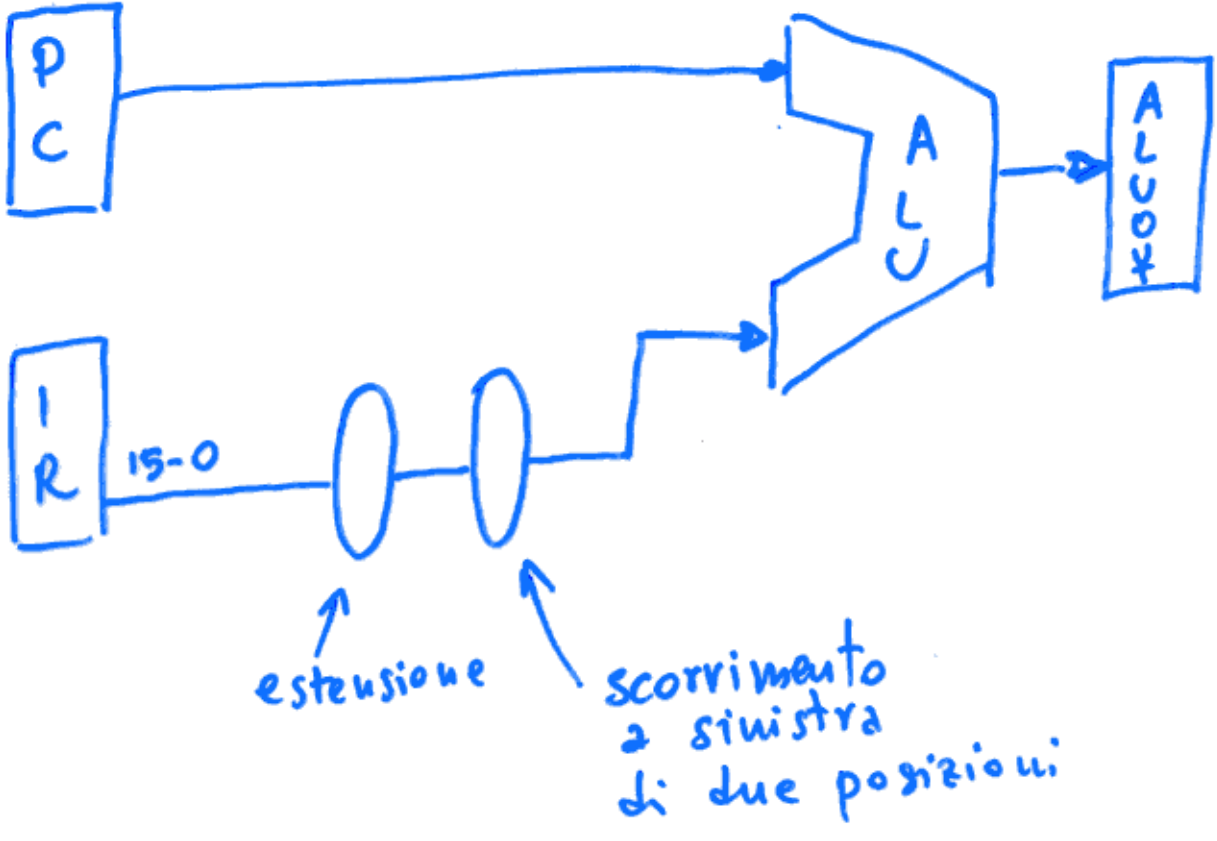
SCA x c) (a)

x effettuare lettura contenuto registri



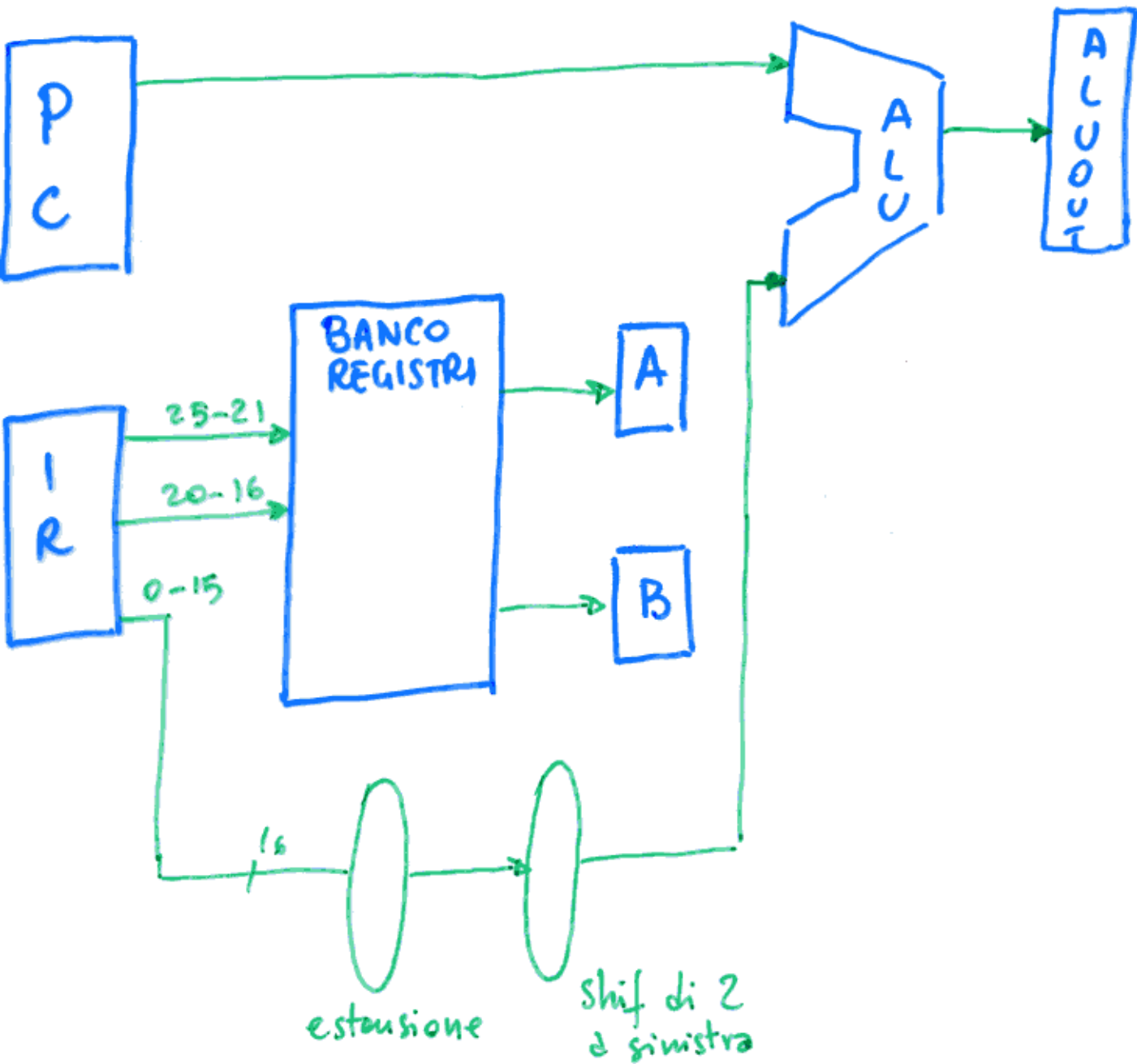
SCA x c) a)

x calcolare eventuale indirizzo successivo



2° passo

SCA COMPLETA

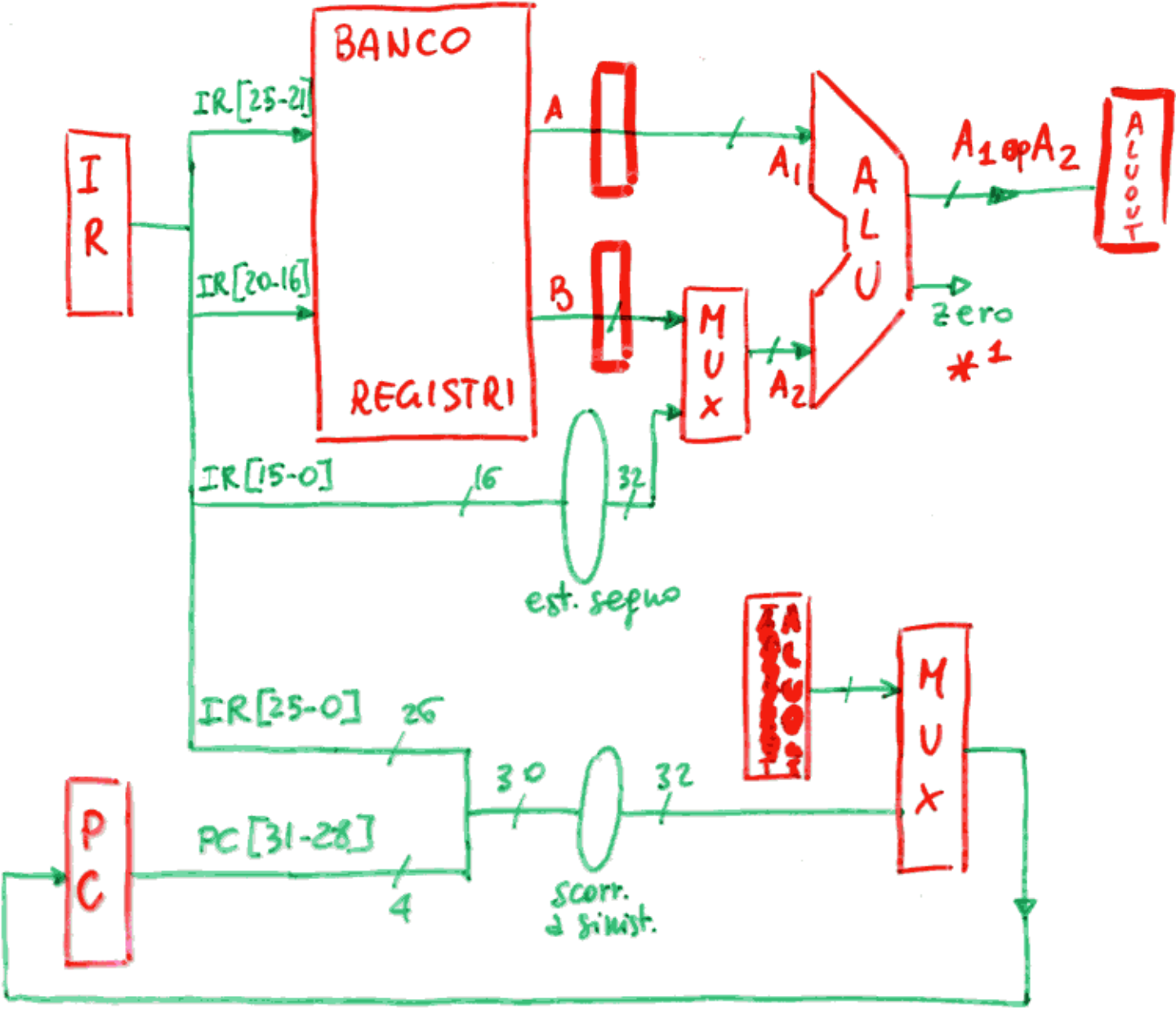


3° PASSO: operazione ALU e completamento esecuzione istruzioni di salto

attività tipo istruzione

- R : $ALU_{OUTPUT} = A \text{ op } B$
- lw/sw : $ALU_{OUTPUT} = A + \text{estens. in segno } IR[15\text{-}0]$
- beq : $ALU_{OUTPUT} = \begin{cases} zero & \text{if } A = B \\ \overline{zero} & \text{if } A \neq B \end{cases}$

  if zero PC ← ALUOut
- Jump : $PC \leftarrow (PC)_{31\text{-}28} \cdot IR[25\text{-}0] << 2$



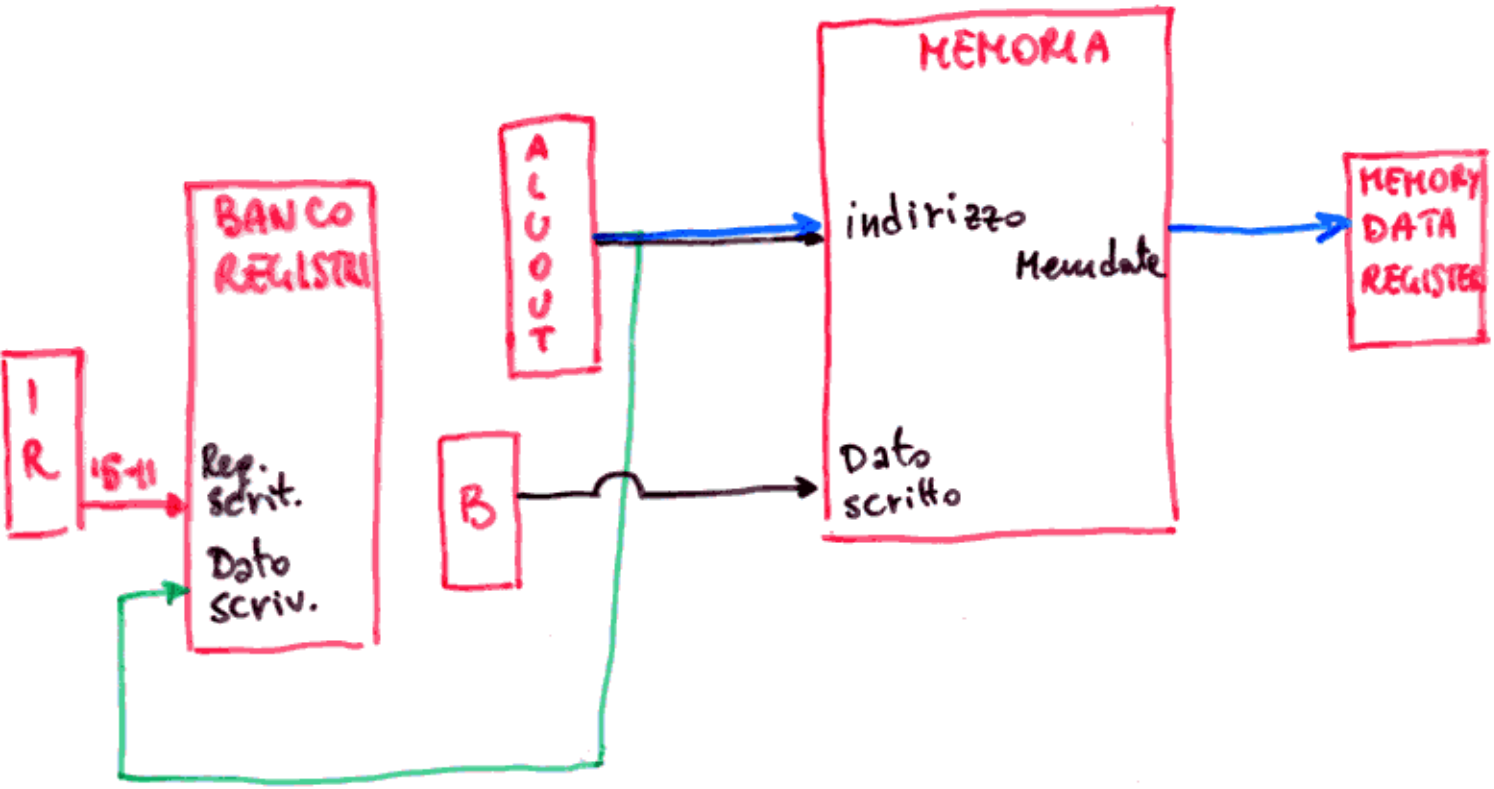
*1 per poter sfruttare questa informazione nello stesso ciclo di CK non la si deve mandare al SCO ma la si deve usare in AND con PCwrite cond. (vedere lucidi successivi)

4° PASSO:

- completamento istruzione di tipo "R"
- accesso in memoria x istruzione "lw"
- completamento istruzione "sw"

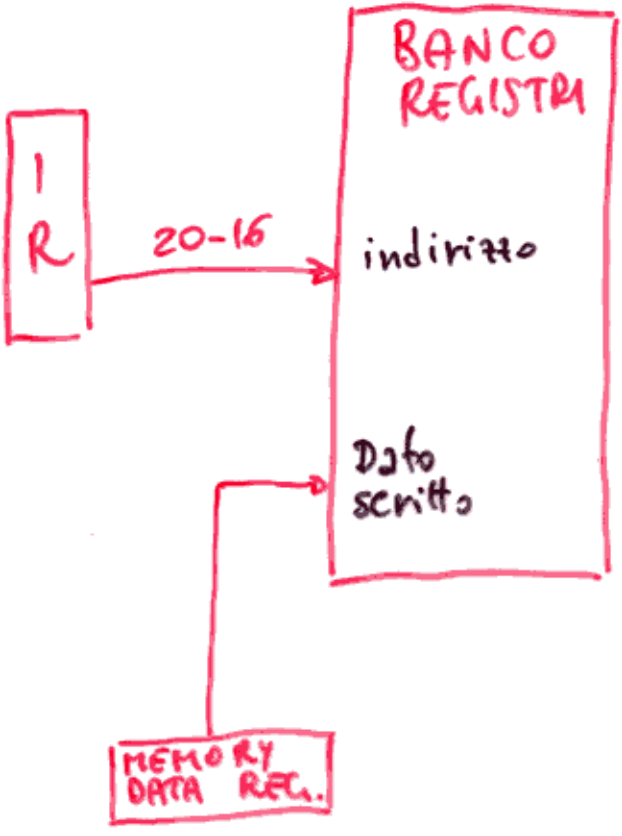
attività x Tipo istruzione o istruzione:

- R: ALUOUT → registro [IR [15-11]]
- lw: MEMORY DATA REGISTER ← MEMORIA [ALUOUT]
- sw: MEMORIA [ALUOUT] ← B

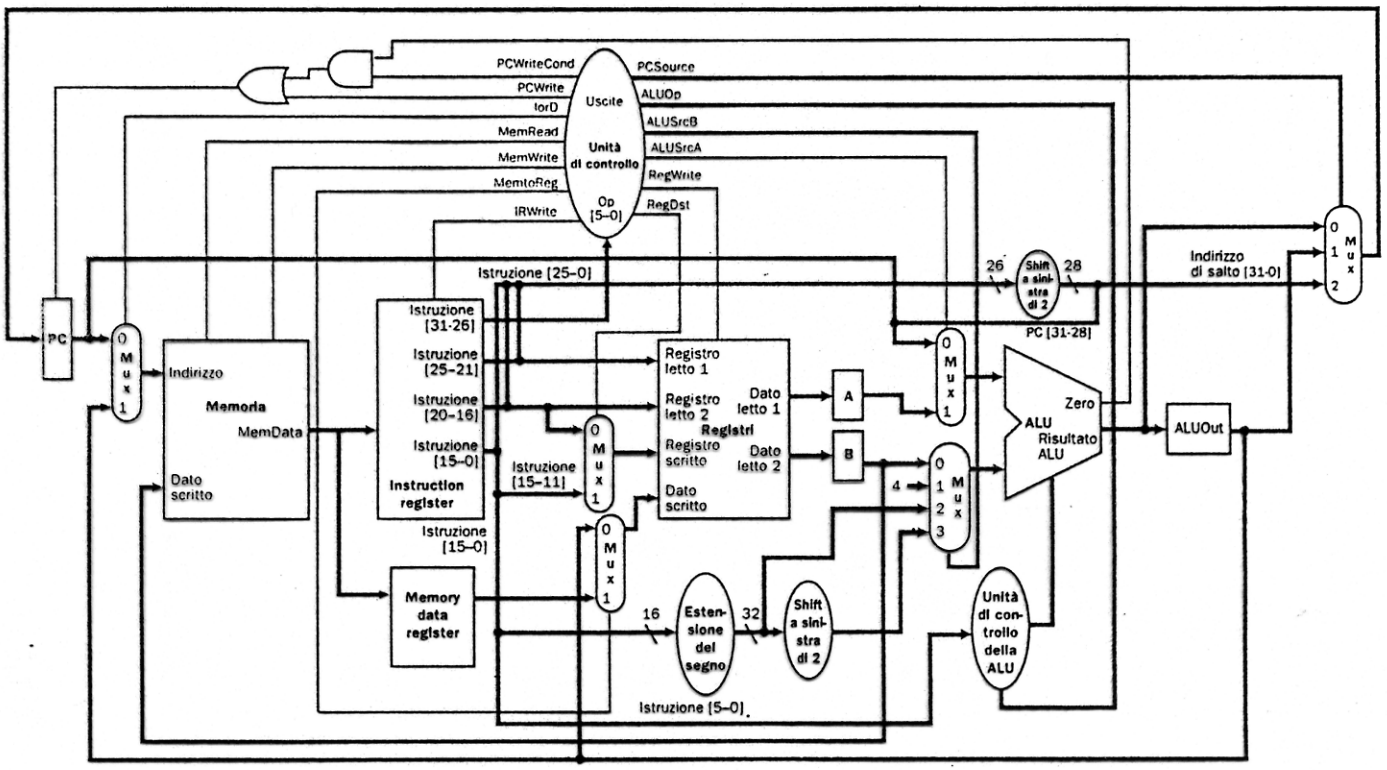


5° PASSO: completamento istruzione "lw"
"fase di riscrittura"

Registro [IR[20-16]] ← MEMORY DATA REGISTER



ARCHITETTURA SCO-SCA



gura 5.33 L'unità di elaborazione completa ed i relativi segnali di controllo nell'implementazione multi-ciclo. I segnali di controllo della figura 5.32 son nnessi all'unità di controllo e sono stati inclusi tutti gli elementi necessari per la modifica di PC. Le aggiunte principali rispetto alla figura 5.32 sono il multiplexe ilizzato per selezionare la fonte del nuovo valore di PC (in alto a destra), due porte logiche utilizzate per combinare i segnali di scrittura di PC (in alto a sinistra i segnali di controllo PCSource, PCWrite e PCWriteCond. Il segnale PCWriteCond è posto in AND con l'uscita Zero della ALU per decidere se il salto condiziona debba essere eseguito; il risultato è posto in OR con il segnale di controllo PCWrite per generare l'effettivo segnale di controllo per la scrittura di PC. Inoltre, l'u ta di IR è manipolata in modo da inviare i 26 bit inferiori (l'indirizzo del salto incondizionato) alla logica utilizzata per la scelta del nuovo PC: i 26 bit sono scalat inistra di due posizioni, aggiungendo due bit pari a 0 nella parte bassa ed i 28 bit risultanti sono concatenati con i 4 bit più significativi di PC già incrementato.

Azioni dei segnali di controllo di 1 bit

Nome del segnale	Effetto quando non affermato	Effetto quando affermato
RegDst	Il numero del registro destinazione per Registro scritto proviene dal campo rt.	Il numero del registro destinazione per Registro scritto proviene dal campo rd.
RegWrite	Nessuno.	Nel registro specificato sull'ingresso Registro scritto è scritto il valore presente sull'ingresso Dato scritto.
ALUSrcA	Il primo operando della ALU è PC.	Il primo operando della ALU è il registro A.
MemRead	Nessuno.	Il contenuto della cella di memoria determinata dall'ingresso Indirizzo è posto sull'uscita MemData.
MemWrite	Nessuno.	Il contenuto della cella di memoria determinata dall'ingresso Indirizzo è sostituito dal valore presente sull'ingresso Dato scritto.
MemtoReg	Il valore inviato all'ingresso Dato scritto dei registri proviene da ALUOut.	Il valore inviato all'ingresso Dato scritto dei registri proviene da MDR.
IorD	L'indirizzo fornito alla memoria proviene da PC.	L'indirizzo fornito alla memoria proviene da ALUOut.
IRWrite	Nessuno.	L'uscita della memoria viene scritta in IR.
PCWrite	Nessuno.	PC viene scritto; la provenienza del valore è controllata da PCSource.
PCWriteCond	Nessuno.	PC viene scritto se anche l'uscita Zero della ALU è attiva.

Azioni dei segnali di controllo di 2 bit

Nome del segnale	Valore	Effetto
ALUOp	00	La ALU calcola una somma.
	01	La ALU calcola una sottrazione.
	10	La ALU calcola l'operazione determinata dal campo funct dell'istruzione.
ALUSrcB	00	Il secondo ingresso della ALU proviene dal registro B.
	01	Il secondo ingresso della ALU è la costante 4.
	10	Il secondo ingresso della ALU è il valore dei 16 bit meno significativi di IR, estesi a 32 bit con segno.
	11	Il secondo ingresso della ALU è il valore dei 16 bit meno significativi di IR, estesi a 32 bit con segno e scalati a sinistra di 2 bit.
PCSource	00	In PC viene scritta l'uscita della ALU (PC+4).
	01	In PC viene scritto il contenuto di ALUOut (l'indirizzo di destinazione del salto condizionato).
	10	In PC viene scritto l'indirizzo di destinazione del salto incondizionato (IR[25-0] scalato a sinistra di 2 bit e concatenato con PC+4[31-28]).

Figura 5.34 Azioni causate dai valori dei segnali di controllo della figura 5.33. La tabella superiore descrive i segnali di controllo di 1 bit, mentre quella inferiore riporta i segnali di controllo di 2 bit; solamente i segnali di controllo che pilotano i multiplexer possono avere effetto quando non affermati. Queste informazioni sono simili a quelle di figura 5.18 per l'unità di elaborazione a singolo ciclo, ma vi sono parecchi nuovi segnali (IRWrite, PCWrite, PCWriteCond, ALUSrcB e PCSource), mentre altri sono stati eliminati o sostituiti (PCSrc, Branch e Jump).

Nome del passo	Azione per istruzioni di tipo-R	Azione per istruzioni di accesso alla memoria	Azione per salti condizionati	Azione per salti incondizionati
Prelievo dell'istruzione	IR = Memoria[PC] PC = PC+4			
Decodifica dell'istruzione/ caricamento dei registri	A = Reg[IR[25-21]] B = Reg[IR[20-16]] ALUOut = PC+(sign-extend(IR[15-0])<<2)			
Esecuzione, calcolo dell'indirizzo, completamento dei salti	ALUOut = A op B	ALUOut = A+sign-extend (IR[15-0])	if (A == B) then PC = ALUOut	PC = PC[31-28] II (IR[25-0]<<2)
Accesso alla memoria o completamento dell'istruzione di tipo-R	Reg [IR[15-11]] = ALUOut	Load: MDR = Memoria[ALUOut] oppure Store: Memoria[ALUOut] = B		
Completamento della lettura da memoria		Load: Reg[IR[20-16]] = MDR		

Figura 5.35 Schema riassuntivo dei passi intrapresi durante l'esecuzione delle diverse classi di istruzioni. Le istruzioni possono richiedere da tre a cinque passi di esecuzione: i primi due passi sono indipendenti dalla classe di istruzioni, mentre in quelli successivi l'istruzione richiede da uno a tre cicli aggiuntivi per essere completata, a seconda della classe. Le caselle vuote nei passi di accesso alla memoria e di completamento della lettura indicano che le classi di istruzioni corrispondenti richiedono un numero minore di cicli: in un'implementazione multi-ciclo, si inizia l'esecuzione di una nuova istruzione non appena l'istruzione corrente viene completata, quindi tali cicli non sono di attesa e non vengono sprecati. Come già citato, il register file di fatto viene letto ad ogni ciclo, ma finché IR non cambia i valori letti sono sempre uguali; in particolare, il valore letto nel registro B durante lo stadio di decodifica dell'istruzione per le istruzioni branch o di tipo-R è uguale al valore che B assume durante il passo di esecuzione e che viene usato nell'accesso alla memoria per l'istruzione store word.

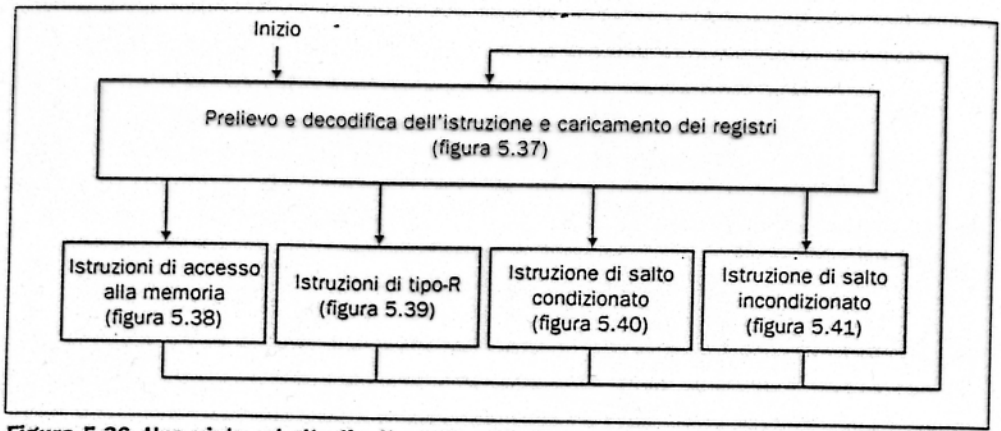


Figura 5.36 Una vista ad alto livello dell'unità di controllo come macchina a stati finiti. I primi passi sono indipendenti dalla classe dell'istruzione, in seguito il completamento delle istruzioni delle diverse classi richiede delle sequenze dipendenti dal codice operativo; dopo aver terminato le azioni richieste per le diverse classi di istruzioni, l'unità di controllo inizia a prelevare una nuova istruzione. Ciascun rettangolo della figura rappresenta uno o più stati; l'arco etichettato con *Inizio* indica lo stato in cui inizia l'esecuzione al momento di prelevare la prima istruzione.

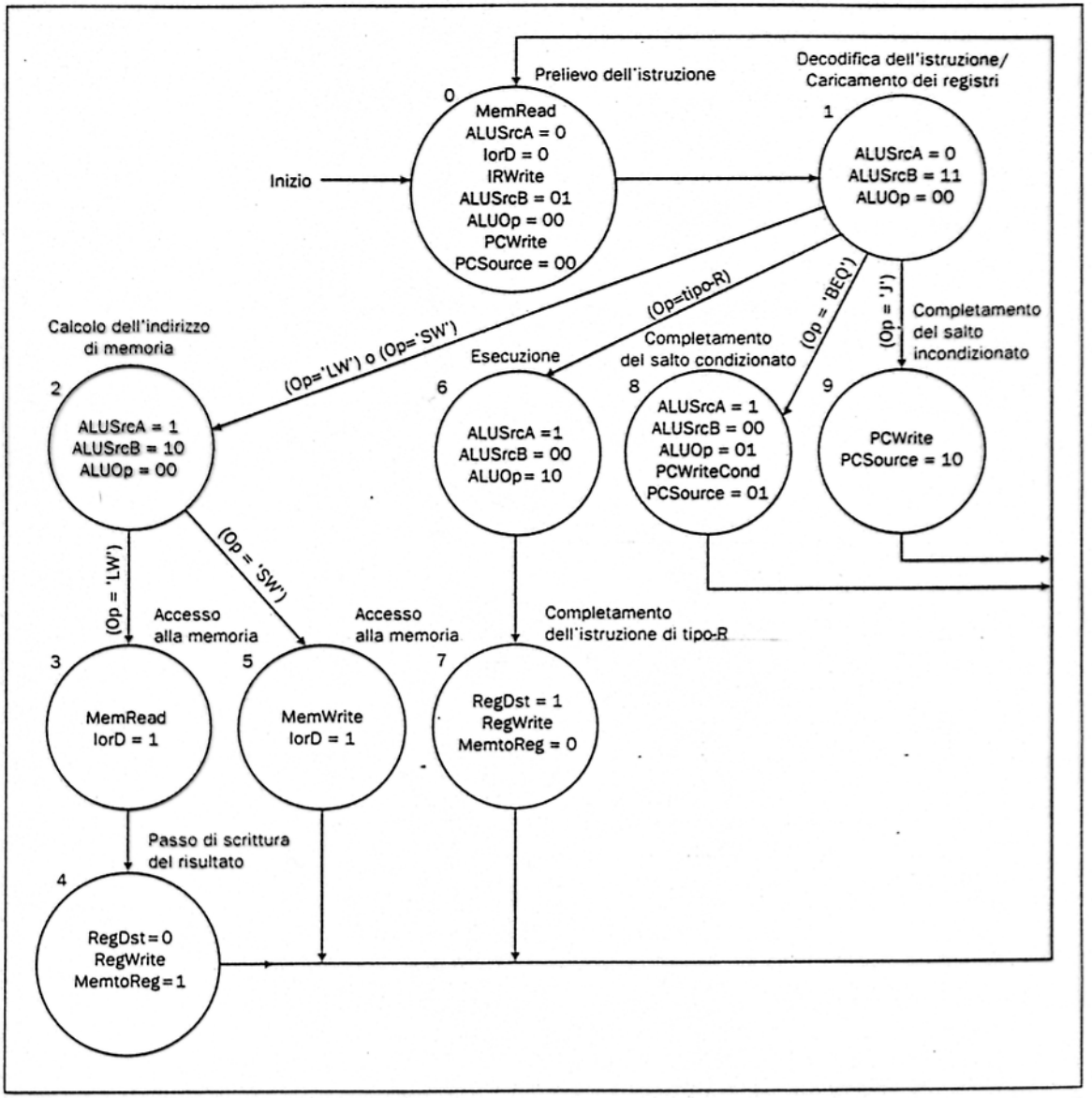


Figura 5.42 La macchina a stati finiti completa dell'unità di controllo relativa all'unità di elaborazione di figura 5.33. Le etichette sugli archi sono le condizioni da verificare nella determinazione dello stato futuro; nel caso in cui tale stato futuro non dipenda da una condizione, l'etichetta è assente. Le etichette interne ai nodi indicano invece i segnali di uscita dell'unità di controllo affermati durante lo stato corrispondente; gli ingressi dei multiplexer vengono sempre specificati quando la loro uscita è utilizzata, per cui in alcuni stati si troveranno ingressi di multiplexer posti a 0. Nell'Appendice C si vedrà come tradurre questa macchina a stati finiti in equazioni logiche e come implementare tali equazioni.

INTERRUZIONI - ECCEZIONI
e loro gestione

CLASSIFICAZIONE EVENTI

TIPO	SORGENTE	TERMINOLOGIA MIPS
RICHIESTA DISPOSITIVO I/O	esterna	INTERRUZIONE
CHIAMATA DEL S.O DA PROGRAMMA UTENTE	interna	ECCEZIONE SUPER VISOR CALL
TRABOCCAMENTO ARITMETICO	interna	ECCEZIONE TRAP o TRAPPOLA
UTILIZZO ISTRUZIONE NON DEFINITA	interna	ECCEZIONE
MALFUNZIONAMENTO HW	entrambe	ECCEZIONE/INTERRUZIONE

GESTIONE INTERRUZIONI

vedere PD32

GESTIONE ECCEZIONI

Exception Program Counter - EPC
indirizzo dell'istruzione che ha causato l'eccezione

FASI

① RIVELAZIONE ECCEZIONE

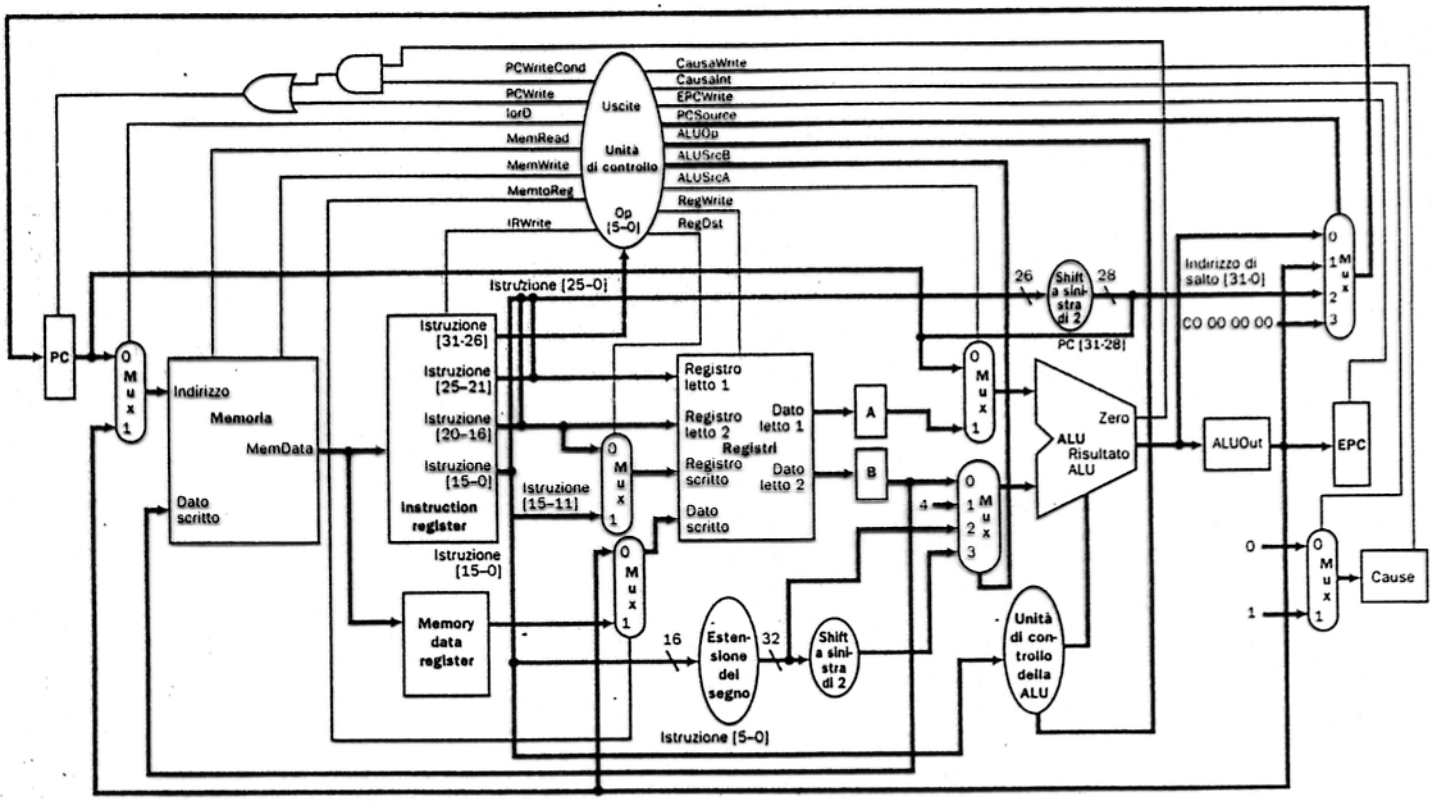
② SALVATAGGIO IND. ISTRUZ. CORRENTE IN EPC

③ TRASFERIRE CONTROLLO AL GESTORE DELLE ECCEZIONI

→ identif. eccez.

- registro CAUSE
- "interruzione" vettorizzata

scelta del MIPS



gura 5.48 L'unità di elaborazione multi-ciclo con le aggiunte necessarie per la gestione delle eccezioni. Le aggiunte comprendono in particolare istri Causa ed EPC, un multiplexer per determinare il valore scritto nel registro Causa, un'espansione del multiplexer che determina il valore da scrivere i ed i segnali di controllo per tali registri e multiplexer.

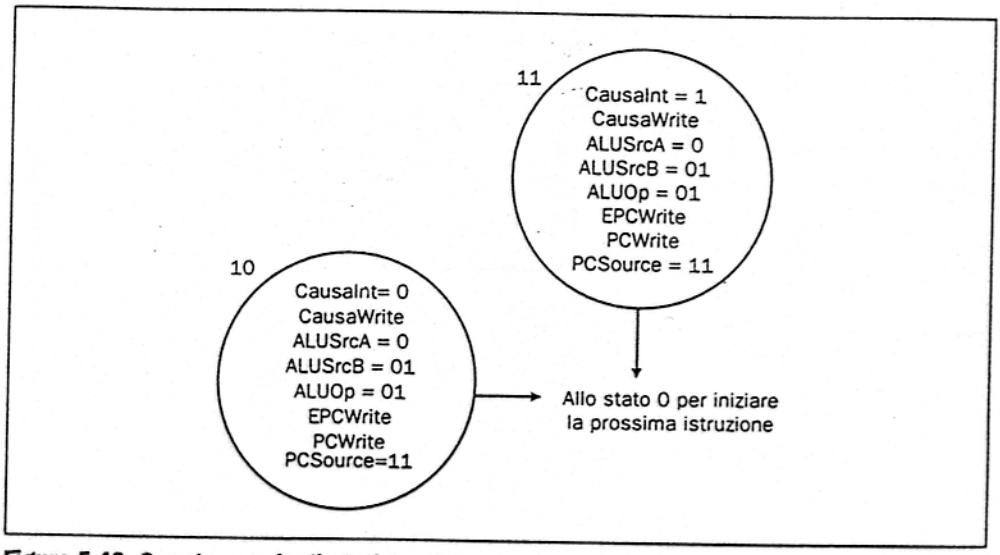


Figura 5.49 Questa coppia di stati gestisce le azioni necessarie per le due eccezioni considerate. Ciascuno stato fornisce il controllo di tre azioni: scrittura di un valore nel registro Causa, memorizzazione dell'indirizzo dell'istruzione responsabile dell'eccezione in EPC e caricamento di PC con l'indirizzo di gestione delle eccezioni. Gli stati 10 e 11 rappresentano l'inizio della gestione delle eccezioni: quando si verifica un'eccezione si trasferisce il controllo ad uno di questi stati. Dopo il completamento dello stato 10 o 11 si restituisce il controllo allo stato 0 iniziando il prelievo di una nuova istruzione.

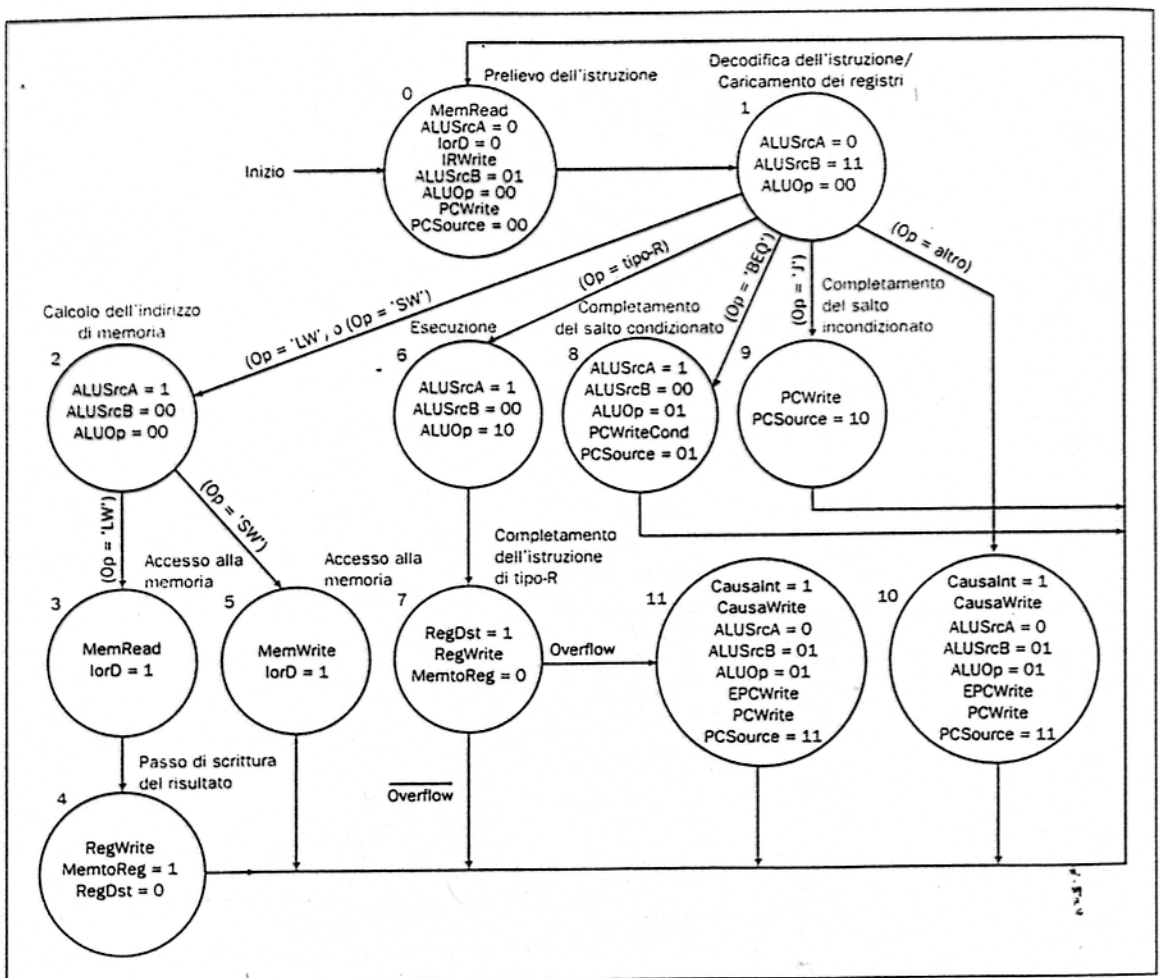
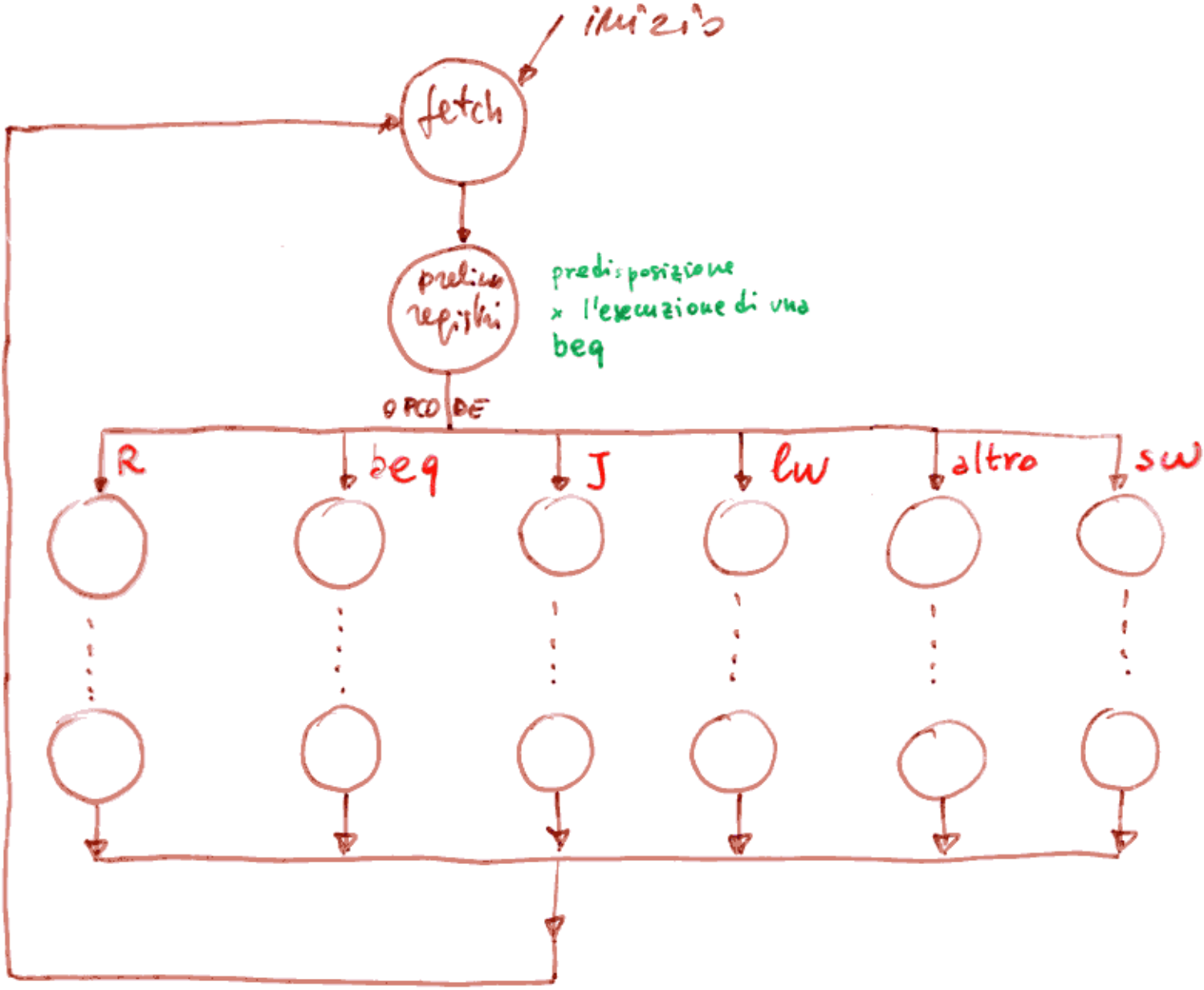


Figura 5.50 La macchina a stati finiti dell'unità di controllo con le aggiunte per il rilevamento e la gestione delle eccezioni. Gli stati 10 ed 11 provengono dalla figura 5.49; l'arco di uscita dello stato 1 etichettato con (*Op=altro*) indica lo stato futuro quando l'ingresso non corrisponde a nessun opcode tra `lw`, `sw`, 0 (tipo-R), `j` e `beq`; l'arco uscente dallo stato 7 etichettato *Overflow* indica l'azione quando la ALU segnala un overflow.

IMPLEMENTAZIONE MICROPROGRAMMATA del SCO



ESERCIZIO:

SCRIVERE IL MICROPROGRAMMA USANDO LO SCHEMA DI MOORE

MULTIMICRO PROGRAMMI