ISTRUZIONI E PSEUDO-ISTRUZIONI MIPS

ARITMETICA

add	$1, $2, $3	$1 := $2 + $3	addizione
addu	$1, $2, $3	$1 := $2 + $3	addizione naturale
addi	$1, $2, cost	$1 := $2 + cost	addizione di costante
addiu	$1, $2, cost	$1 := $2 + cost	addizione naturale di costante
sub	$1, $2, $3	$1 := $2 – $3	sottrazione
subu	$1, $2, $3	$1 := $2 – $3	sottrazione naturale
mult	$1, $2	hi|lo := $1 × $2	moltiplicazione (risultato a 64 bit)

ARITMETICA – pseudo-istruzioni

subi	$1, $2, cost	$1 := $2 – cost	sottrazione di costante
subiu	$1, $2, cost	$1 := $2 – cost	sottrazione naturale di costante
neg	$1, $2	$1 := –$2	negazione aritmetica (compl. a 2)

CONFRONTO

slt	$1, $2, $3	**if** $2 < $3 **then** $1 := 1 **else** $1 := 0	poni a 1 se strettamente minore
sltu	$1, $2, $3	**if** $2 < $3 **then** $1 := 1 **else** $1 := 0	poni a 1 se strettamente minore naturale
slti	$1, $2, cost	**if** $2 < cost **then** $1 := 1 **else** $1 := 0	poni a 1 se strettamente minore di costante
sltiu	$1, $2, cost	**if** $2 < cost **then** $1 := 1 **else** $1 := 0	poni a 1 se strettamente minore di costante naturale

LOGICA

or	$1, $2, $3	$1 := $2 **or** $3	somma logica bit a bit
and	$1, $2, $3	$1 := $2 **and** $3	prodotto logico bit a bit
ori	$1, $2, cost	$1 := $2 **or** cost	somma logica bit a bit con costante
andi	$1, $2, cost	$1 := $2 **and** cost	prodotto logico bit a bit con costante
nor	$1, $2, $3	$1 := $2 **nor** $3	somma logica negata bit a bit
sll	$1, $2, cost	$1 := $2 << cost	scorrimento logico a sinistra (left) del numero di bit specificato da cost
srl	$1, $2, cost	$1 := $2 >> cost	scorrimento logico a destra (right) del numero di bit specificato da cost

LOGICA – pseudo-istruzione

not	$1, $2	$1 := **not** $2	negazione logica (not bit a bit)

SALTO INCONDIZIONATO: ASSOLUTO, INDIRETTO E CON COLLEGAMENTO

j	indir	pc := indir (28 bit)	salto incondizionato assoluto
jr	$1	pc := $1 (32 bit)	salto incondizionato indiretto da registro
jal	indir	pc := indir (28 bit) e collega il registro $ra	salto incondizionato assoluto con collegamento

SALTO CONDIZIONATO

beq	$1, $2, spi	**if** $1 = $2 salta relativo a PC	salto condizionato di uguaglianza
bne	$1, $2, spi	**if** $1 ≠ $2 salta relativo a PC	salto condizionato di disuguaglianza

SALTO CONDIZIONATO – pseudo-istruzioni

blt	$1, $2, spi	**if** $1 < $2 salta relativo a PC	salta se strettamente minore
bgt	$1, $2, spi	**if** $1 > $2 salta relativo a PC	salta se strettamente maggiore
ble	$1, $2, spi	**if** $1 ≤ $2 salta relativo a PC	salta se minore o uguale
bge	$1, $2, spi	**if** $1 ≥ $2 salta relativo a PC	salta se maggiore o uguale

TRASFERIMENTO TRA PROCESSORE E MEMORIA

lw	$1, spi ($2)	$1 := mem ($2 + spi)	carica parola (a 32 bit)
sw	$1, spi ($2)	mem ($2 + spi) := $1	memorizza parola (a 32 bit)
lh, *lhu*	$1, spi ($2)	$1 := mem ($2 + spi)	carica mezza parola (a 16 bit)
sh	$1, spi ($2)	mem ($2 + spi) := $1	memorizza mezza parola (a 16 bit)
lb, *lbu*	$1, spi ($2)	$1 := mem ($2 + spi)	carica byte (a 8 bit)
sb	$1, spi ($2)	mem ($2 + spi) := $1	memorizza byte (a 8 bit)

TRASFERIMENTO TRA PROCESSORE E MEMORIA – pseudo-istruzioni (vedi nota 1 sotto)

lw	$1, etichetta	$1 := mem ($gp + spi di etichetta)	carica parola (a 32 bit)
sw	$1, etichetta	mem ($gp + spi di etichetta) := $1	memorizza parola (a 32 bit)

TRASFERIMENTO TRA REGISTRI (non referenziabili)

mflo	$1	$1 := lo	copia registro lo
mfhi	$1	$1 := hi	copia registro hi

TRASFERIMENTO TRA REGISTRI – pseudo-istruzione

move	$1, $2	$1 := $2	copia registro

CARICAMENTO DI COSTANTE IN REGISTRO

lui	$1, cost	$1 (16 bit più signif.) := cost	carica cost (in 16 bit più signif. di $1) (16 bit meno signif. di $1 posti a 0)

CARICAMENTO DI COSTANTE / INDIRIZZO IN REGISTRO – pseudo-istruzioni (vedi nota 2 sotto)

li	$1, cost	$1 := cost (32 bit)	carica costante a 32 bit
la	$1, indir	$1 := indir (32 bit)	carica indirizzo a 32 bit

REGISTRI MIPS

REGISTRI REFERENZIABILI

0	*0*	costante 0 (registro denotabile anche come $zero)
1	*at*	uso riservato all'assembler-linker (per espandere pseudo-istruzioni e macro)
2 - 3	*v0 - v1*	valore restituito da funzione (sottoprogramma) (*v0* per dati di tipo scalare o puntatore, più *v1* per numeri reali di tipo *double*)
4 - 7	*a0 - a3*	argomenti (scalari o punt) in ingresso a funzione (max quattro argomenti)
8 - 15	*t0 - t7*	registri per valori temporanei (p. es. calcolo delle espressioni)
16 - 23	*s0 - s7*	registri usabili (se possibile) per var locali (scalari o punt) di sottoprogramma
24 - 25	*t8 - t9*	registri per valori temporanei (in aggiunta a *t0 - t7*), come i precedenti *tx*
26 - 27	*k0 - k1*	registri riservati per il nucleo (kernel) del Sistema Operativo
28	*gp*	global pointer (puntatore all'area dati globale)
29	*sp*	stack pointer (puntatore alla cima della pila)
30	*fp*	frame pointer (puntatore all'area di attivazione di sottoprogramma)
31	*ra*	return address (indirizzo di rientro da chiamata a sottoprogramma)

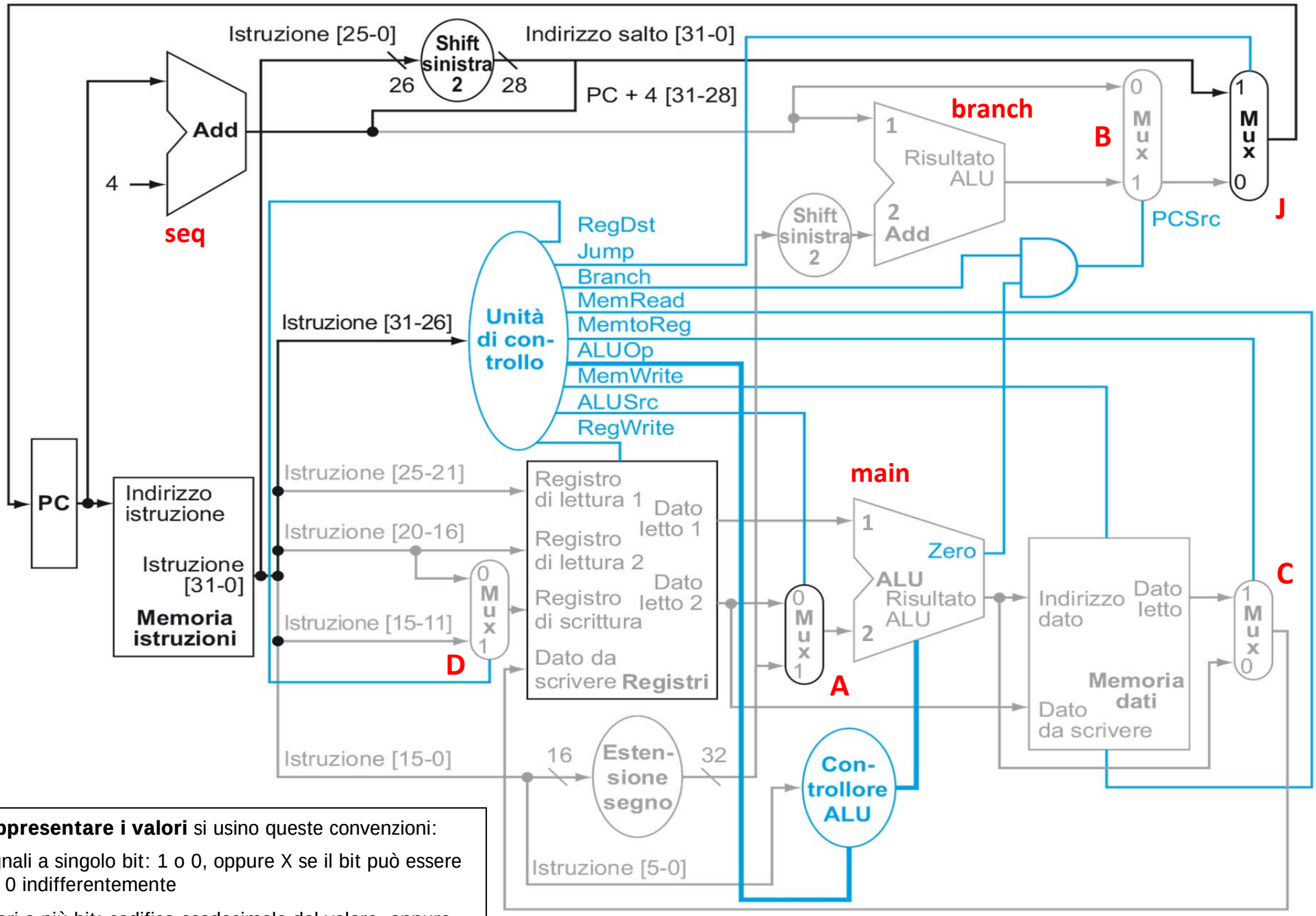
REGISTRI NON REFERENZIABILI

	pc	program counter (contatore di programma)
	hi	registro per risultato di moltiplicazione e divisione (32 bit più significativi)
	lo	registro per risultato di moltiplicazione e divisione (32 bit meno significativi)

Nota 1: anche le pseudo-istruzioni *lh*, *lhu*, *sh*, *lb*, *lbu* e *sb* con etichetta sottintendono il registro $gp.

Nota 2: entrambe le pseudo-istruzioni *li* e *la* caricano un valore a 32 bit in un registro, ma *li* va usata per caricare una costante e *la* per caricare un indirizzo.

PROCESSORE SINGOLO CICLO (monociclo)

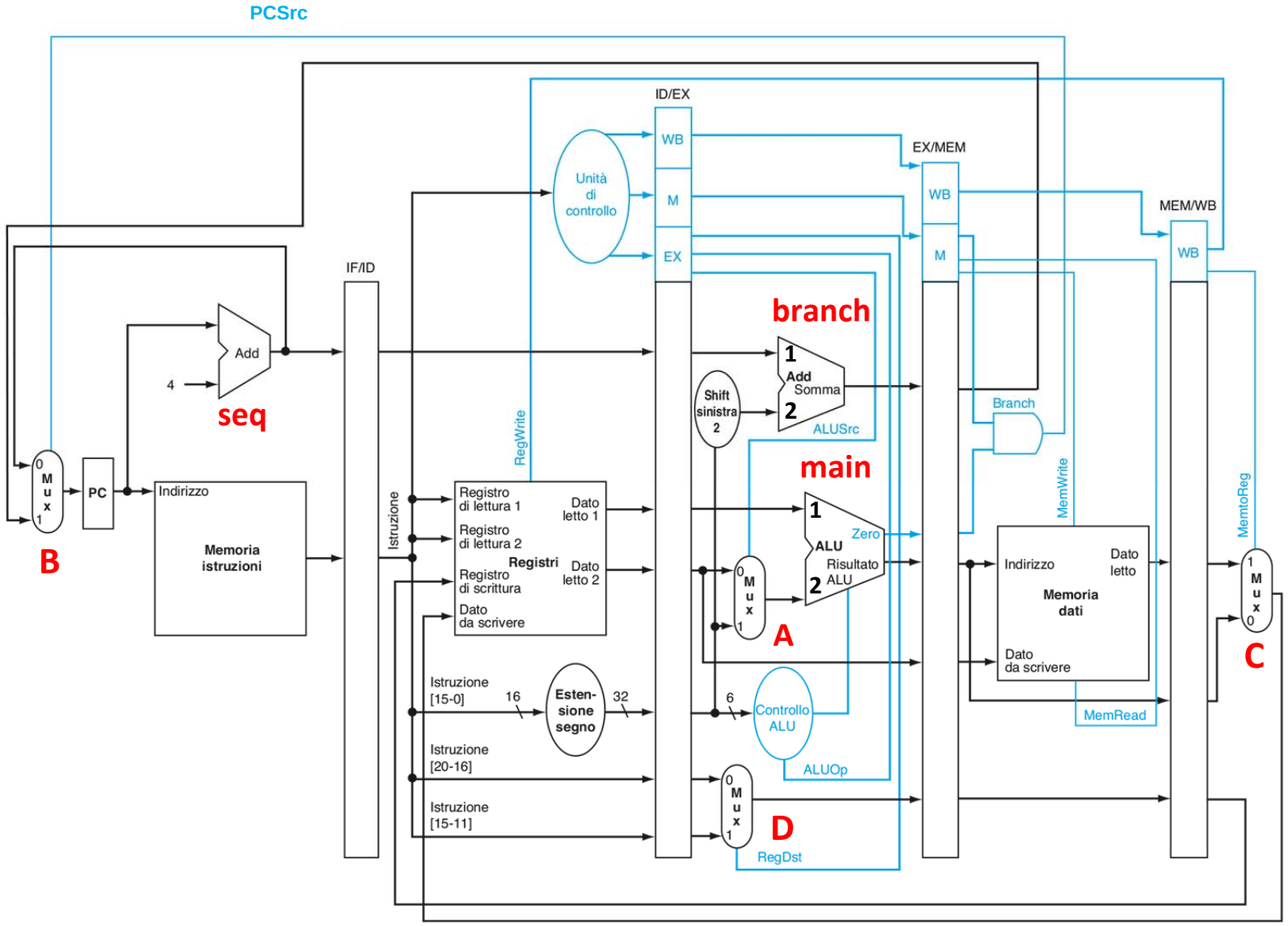


Per **rappresentare i valori** si usino queste convenzioni:

- segnali a singolo bit: 1 o 0, oppure X se il bit può essere 1 o 0 indifferentemente
- valori a più bit: codifica esadecimale del valore, oppure "n.d." se valore non determinabile per il caso richiesto

in grassetto i nomi dei multiplexer (**A**, **B**, **C**, **D** e **J**) e delle ALU (**main**, **branch** e **seq**)

PROCESSORE PIPELINE (multiciclo) e FORMATI DEI REGISTRI INTER-STADIO



Per **rappresentare i valori** si usino queste convenzioni:

- segnali a singolo bit: 1 o 0, oppure X se il bit può essere 1 o 0 indifferentemente
- valori a più bit: codifica esadecimale del valore, oppure “n.d.” se valore non determinabile per il caso richiesto

Legenda dei registri inter-stadio

(32) ecc. = n° di bit del campo considerato

(Rs) o (Rt) = contenuto del registro Rs o Rt

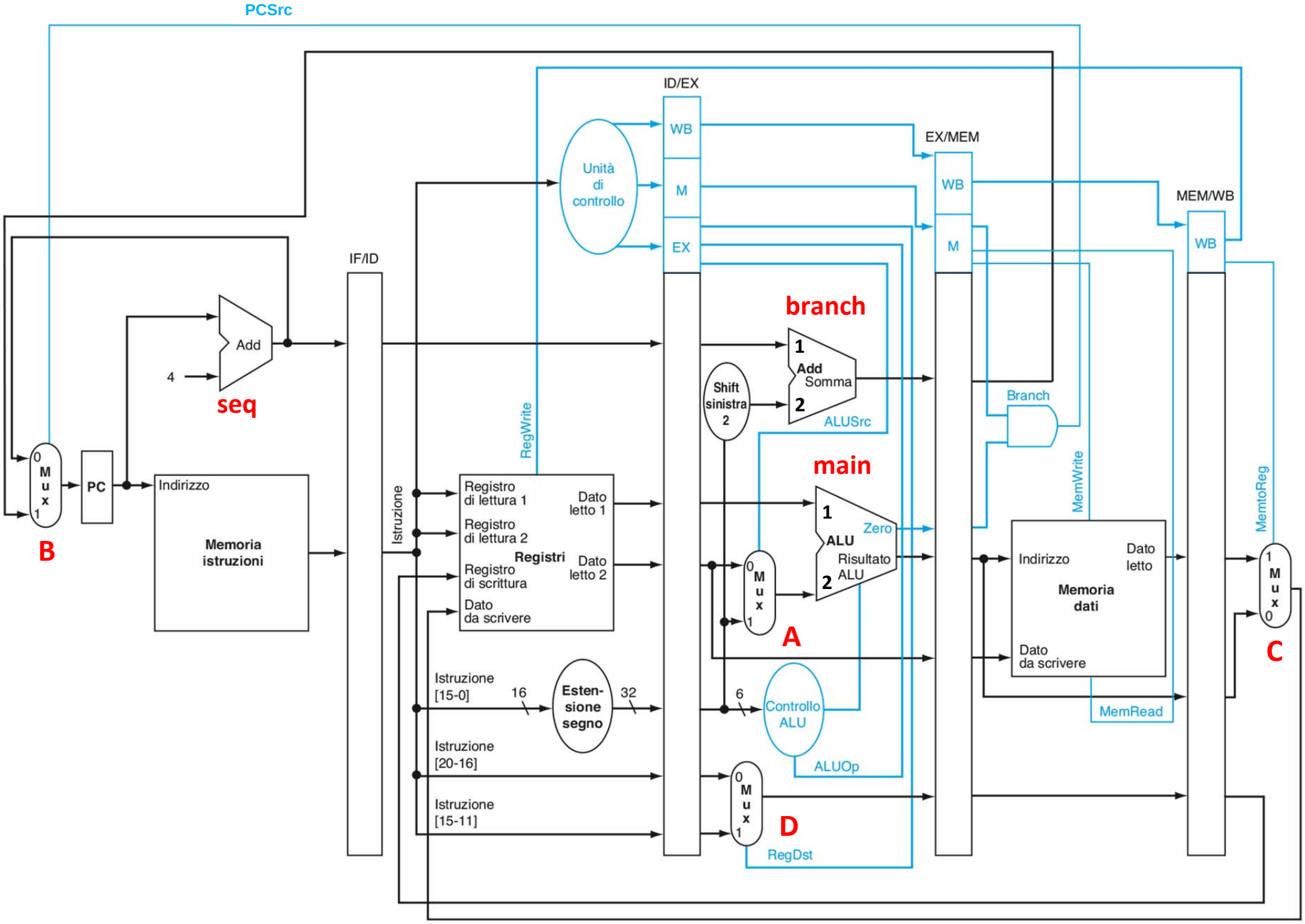
Rd, Rt, R = numero di registro, dove R può essere Rd o Rt

I campi dei registri inter-stadio sono indicati come: Nome_Registro_Interstadio.Campo_x (p. es. MEM/WB.dato letto)

IF/ID	PC (32)	istruzione (32)							
ID/EX	WB (2)	M (3)	EX (4)	PC (32)	(Rs) (32)	(Rt) (32)	imm/offset esteso (32)	Rt (5)	Rd (5)
EX/MEM	WB (2)	M (3)	PC (32)	ALU_out (32)	bit Z (1)	(Rt) (32)	R (5)		
MEM/WB	WB (2)	dato letto (32)	ALU_out (32)	R (5)					

formati dei registri inter-stadio

PROCESSORE PIPELINE (multiciclo) ingrandito per comodità di lettura



PROCESSORE PIPELINE (multiciclo) con UNITÀ di STALLO e UNITÀ di PROPAGAZIONE (omessi altri dettagli)