

ARCHITETTURA DEGLI ELABORATORI

ESERCIZIO I [8 PUNTI]

I registri $R1$ e $R2$ contengono indirizzi di parole di memoria, mentre il registro $R3$ contiene un dato numerico. In particolare, la parola memorizzata all'indirizzo contenuto in $R1$ contiene un valore che indicheremo con α . Si scriva un frammento di codice assembly che:

- se $\alpha = 0$ raddoppi il registro $R3$ e scriva all'indirizzo contenuto in $R2$ il risultato ottenuto;
- se $\alpha \neq 0$ triplichi il registro $R3$ e scriva all'indirizzo contenuto in $R2$ il risultato ottenuto.

Soluzione

lw R4, 0(R1)	$R4 \leftarrow \alpha$
add R5, R3, R3	$R5 \leftarrow 2R3$
beq R4, 0, ETI	se $\alpha = 0$ salta a ETI
add R5, R5, R3	$R5 \leftarrow 3R3$
ETI: add R3, R5, 0	$R3 \leftarrow R5$
sw R5, 0(R2)	

ESERCIZIO III [11 PUNTI]

Si assuma che il valore inizialmente contenuto nel registro Ri sia $2i$ e che venga eseguito il fetch della seguente istruzione:

$lw\ R5, 32(R2)$

Contestualmente il controllore emette questi segnali:

$$RegDest = 1, AluSrc = 0$$

- (1) Si fornisca, in esadecimale, la codifica dell'istruzione.
- (2) Si scriva, in notazione esadecimale, il valore dato dai 32 bit in uscita dalla ALU; si indichi se tale valore è corretto.
- (3) Si dica quali registri cambiano valore dopo l'esecuzione dell'istruzione; si indichi se tali registri sono quelli corretti.

Soluzione punto 1: La codifica binaria dell'istruzione è:

1000 1101 0100 1101 0000 0000 0010 0000

, quella esadecimale invece è:

8 D 4 D 0 0 2 0

Soluzione punto 2:

A causa del segnale $AluSrc = 0$ la ALU svolge $(rs) + (rt)$ anzichè $(rs) + SE(IR[15 \dots 0])$. Quindi in uscita dalla ALU si ha $(R5) + (R2) = 10 + 4 = 14$, in esadecimale

0 0 0 0 0 0 E

. Il segnale di controllo $AluSrc$ è errato e quindi tale valore non è corretto.

Soluzione punto 3:

A causa del segnale $RegDest = 1$ il registro che viene scritto è rt che corrisponde a $R5$. In questo caso l'esecuzione e il valore del segnale di controllo $RegDest$ sono corretti.

INFO UTILI

istruzione	codice	function	ALUOp
and	000000	100100	00
or	000000	100101	00
add	000000	100000	00
sub	000000	100010	00
slt	000000	101010	00
sw	101011	xxxxxx	01
lw	100011	xxxxxx	01
beq	000100	xxxxxx	10
nop	000000	000000	00