CALCOLATORI ELETTRONICI – PROVA II – 14/6/2016 – II FACOLTA' DI INGEGNERIA (CESENA)

COGNOME	NOME	MATRICOLA	PARI/DISPARI	POSTO

TESTO - Tempo disponibile: 70 minuti

Prima domanda introduttiva al compito (Punti 5)

1. Facendo riferimento all'architettura del DLX vista a lezione si dica come i bit del registro IR (Instruction Register) della U.d.C. vengono collegati ai decoder del *register file* nelle istruzioni di tipo **R** e nella istruzione di **LOAD** (che è una istruzione di tipo I)

Si vuole ora estendere l'ISA del DLX visto a lezione con le istruzioni di accesso a uno STACK localizzato in memoria principale (cioè indirizzato tramite MAR). A tal fine si assegna al registro R30 la funzione di Stack Pointer (SP) e si introducono le seguenti due istruzioni:

- PUSH Ri
- POP Ri

Si adottano poi le seguenti convenzioni:

- Lo stack cresce verso gli indirizzi decrescenti
- Lo SP punta all'ultimo byte occupato (cioè al byte meno significativo dell'ultima word salvata sullo stack)

Inoltre si assume che SP (cioè R30) sia stato inizializzato con un valore multiplo di 4 (si assume cioè che gli elementi dello stack sono allineati).

L'istruzione **PUSH Ri** è quindi realizzata con le seguenti operazioni descritte in RTL:
R30 ← R30-4
M[R30] ← Ri

L'istruzione **POP Ri** è quindi realizzata con le seguenti operazioni descritte in RTL:
Ri ← M[R30]
R30 ← R30+4

2. Si ipotizzi che il codice operativo della PUSH sia 34H e il codice operativo della POP sia 36H, e si mostri la codifica delle nuove istruzioni scegliendo uno dei 3 formati R, I, J delle istruzioni del DLX, e avendo cura di indicare come il registro IR della U.d.C. dovrà essere collegato ai decoder del register file: si devono modificare queste connessioni rispetto a quelle individuate nella risposta alla domanda 1? Sono necessarie altre connessioni tra IR e datapath? **(punti 5)**

3. Con riferimento al datapath del DLX sequenziale visto a lezione, si disegni ora il diagramma degli stati che controlla l'esecuzione di una delle due nuove istruzioni utilizzando il minor numero di stati possibile, inserendo anche gli stati necessari alle fasi di fetch e decodifica. In particolare si chiede agli studenti con numero di matricola **dispari** di scegliere la **PUSH** e a quelli con numero di matricola pari di scegliere la **POP. Quanti periodi di clock sono necessari per eseguire l'istruzione appena progettata nel caso in cui l'accesso alla memoria richieda 2 stati di wait?**

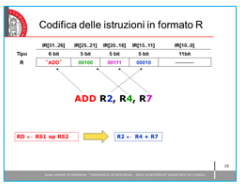
4. Si chiede ora di scrivere nell'ISA del DLX "standard" la sequenza di istruzioni necessaria a eseguire la stessa funzione svolta dalla istruzione appena progettata **(punti 5).**

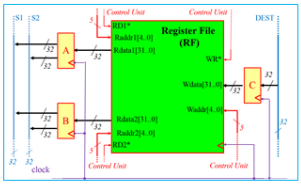
5. Si chiede ora ai soli studenti **con numero di matricola dispari** di indicare come si potrebbe modificare l'architettura del datapath, se si volesse realizzare l'istruzione di PUSH Ri senza utilizzare un registro del register file come Stack Pointer. Quali altri segnali di controllo da UDC a datapath si dovrebbero aggiungere? In questo caso come e perchè si semplificherebbe il formato dell'istruzione?

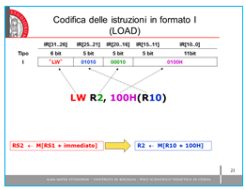
TRACCIA DI SOLUZIONE

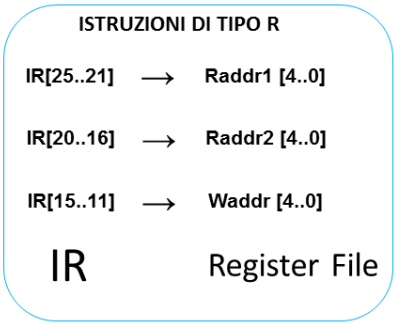
Quesito N.1
Facendo riferimento all'architettura del DLX vista a lezione si dica come i bit del registro IR (Instruction Register) della U.d.C. vengono collegati ai decoder del *register file* nelle istruzioni di tipo **R** e nella istruzione di **LOAD** (che è una istruzione di tipo I)

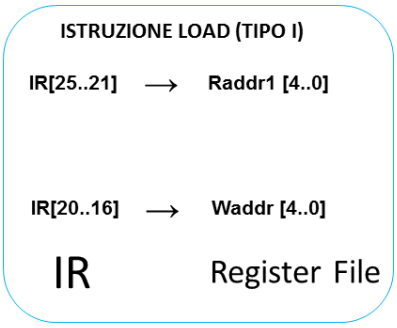
Dalle seguenti tre slide del corso si ricava la risposta sottostante.





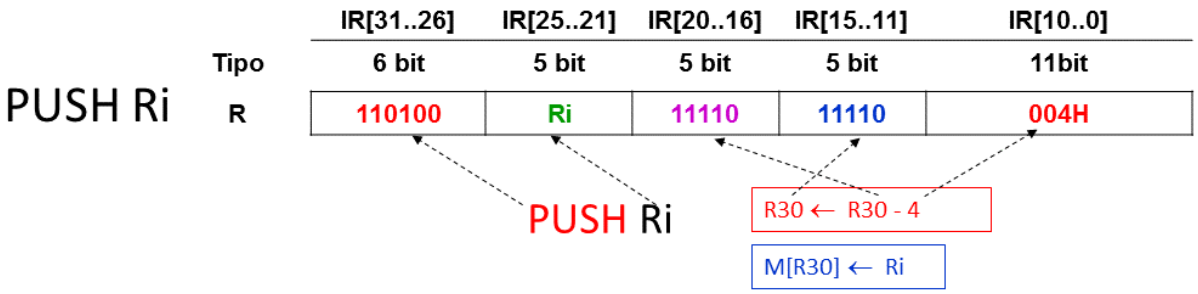




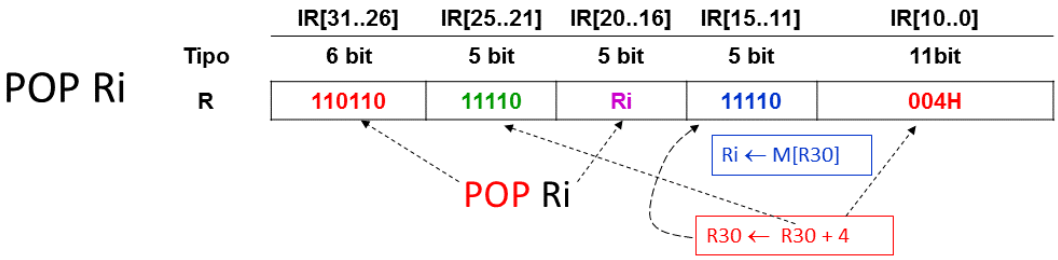


Quesito N. 2
Si ipotizzi che il codice operativo della PUSH sia 34H e il codice operativo della POP sia 36H, e si mostri la codifica delle nuove istruzioni scegliendo uno dei 3 formati R, I, J delle istruzioni del DLX, e avendo cura di indicare come il registro IR della U.d.C. dovrà essere collegato ai decoder del register file: si devono modificare queste connessioni rispetto a quelle individuate nella risposta alla domanda 1? Sono necessarie altre connessioni tra IR e datapath? **(punti 5)**

L'istruzione PUSH Ri ha due registri sorgente (Ri e R30) e un registro di destinazione (R30), quindi scegliamo per la PUSH Ri il formato R; in questo modo possiamo inserire nei bit [10..0] (liberi) la costante 4 che va sottratta da R30. Quindi è possibile definire il formato della PUSH Ri come qualunque istruzione ALU (tipo R) senza dover richiedere alcuna modifica alle interconnessioni IR – Register File utilizzate nelle istruzioni ALU con operandi nei registri



L'istruzione POP Ri ha un registro sorgente (R30) e due registri destinazione (R30 e Ri). Ricordiamo che nelle istruzioni di tipo R il decoder della Write Logic del Register file è pilotato dai bit IR[15..11]. Ricordiamo poi che anche il campo RS2 (cioè IR[20..16]) deve già arrivare alla Write Logic in alcune istruzioni di tipo I (vedi ad esempio la risposta al quesito 1). Si noti che questo presuppone che ci sia un MUX da 5 bit a due vie davanti al decoder della Write Logic! Quindi, per non modificare le interconnessioni tra IR e RF dobbiamo mettere un operando destinazione in Rs2 e uno in Rd; questo implica che il formato dell'istruzione sarà di tipo R, ma nel diagramma degli stati della POP Ri ci troveremo entrambe le seguenti microistruzioni:
RS2 ← C e Rd ← C. Proponiamo quindi di adottare il formato di tipo R anche per l'istruzione POP Ri con i campi Rs1, Rs2 e Rd assegnati come segue:



Nei bit liberi IR[10..0] inseriremo la costante 4 che va sommata a R30.
Se avessimo utilizzato il formato I, avendo due registri di destinazione avremmo dovuto portare anche RS1 al decoder della Write Logic, e quindi avremmo dovuto trasformare il MUX che lo pilota da MUX a due vie a MUX a tre vie!

Quesito n. 4
Si chiede ora di scrivere nell'ISA del DLX "standard" la sequenza di istruzioni necessaria a eseguire la stessa funzione svolta dalle nuove istruzioni proposte **(punti 5).**

Il quesito chiede di scrivere in assembler DLX il codice che realizzi le funzioni di PUSH Ri (se il numero di matricola è dispari) e POP Ri (se il numero di matricola è pari) .

PUSH Ri:

```
SUBI R30, R30, 4
SW 0 (R30), Ri
```

POP Ri:

```
LW Ri, 0 (R30)
ADDI R30, R30, 4
```

Sono dunque necessarie due istruzioni "native" DLX per realizzare ciascuna delle due istruzioni assegnate.
Si suggerisce ora di calcolare il numero di periodi di clock necessario a eseguire ciascuna delle suddette operazioni e lo si confronti con il CPI delle corrispondenti istruzioni aggiunte all'ISA con la risposta al quesito 3.
Il risparmio di tempo (cioè la riduzione del CPUtime) è importante in quanto in generale le operazioni sullo stack hanno frequenza abbastanza elevata (si consideri l'espressione del CPUtime in funzione del numero di istruzioni e del CPImedio).

Quesito 5.
Si chiede ora ai soli studenti **con numero di matricola dispari** di indicare come si potrebbe modificare l'architettura del datapath, se si volesse realizzare l'istruzione di PUSH Ri senza utilizzare un registro del register file come Stack Pointer. Quali altri segnali di controllo da UDC a datapath si dovrebbero aggiungere? In questo caso come e perchè si semplificherebbe il formato dell'istruzione?

Per disporre di uno stack è necessario avere lo stack pointer (SP). Quindi se non possiamo riservarci un registro del RF per svolgere la funzione di SP, possiamo ad esempio aggiungere al data path un ulteriore registro da 32 bit, che si affianchi agli altri; potremmo chiamare questo registro SP o TEMP2 per differenziarlo dagli altri.

L'aggiunta di un registro al data path implica l'aggiunta anche di 3 segnali di controllo: i due output enable (OE1 e OE2) e il write enable (WE).

Il formato dell'istruzione si semplificherebbe rispetto a quello proposto nella risposta al quesito 2 in quanto non sarebbe necessario inserire R30 nei campi dell'istruzione. Lo studente provi a modificare la sua risposta al quesito 3 nell'ipotesi di sostituire R30 con il nuovo registro aggiunto al data path.

COMPLEMENTI DI TEORIA
Si noti che con le aggiunte proposte da questo compito (due istruzioni e un registro con la funzione di stack pointer) potremmo dire che il DLX diventa una architettura dotata di stack. Per sfruttare adeguatamente lo stack nel nesting delle procedure e degli interrupt dovremmo modificare l'implementazione delle istruzioni JumpAndLink nonché quelle di ritorno (da procedura e da interrupt).