DOMANDE TORICHE PARTE 2 COMPITO ARCH ELAB LUGLIO 2011
(le stesse domande sono state fatte al SECONDO COMPITINO 2011)

1. **Si spieghi in dettaglio la divisione fra numeri reali secondo lo standard IEEE 754.**

Soluzione
Nell'effettuare la divisine tra due numeri floating point secondo standard IEEE 754 bisogna seguire i seguenti passaggi:

- controllo se uno o entrambi gli operandi sono zero.
  Se il divisore è zero verrà dato errore, quindi Non A Number, se il dividendo è zero ed il divisore è diverso da zero, verrà dato output zero. In entrambi questi casi il processo di divisione termina.
- l'esponente del divisore viene sottratto all'esponente del dividendo
- nell'esponente risultate dal passo precedente è stata sottratta la polarizzazione, in quanto era stata applicata ad entrambi gli esponenti ed avendone fatto la differenza la polarizzazione è stata persa. La polarizzazione va quindi nuovamente ripristinata. (In pratica all'esponente risultate dal passo precedente viene sottratto il valore 2^{k-1}-1)
  Se non ci sono underflow od overflow di esponente la procedura continua.
- avviene la divisione degli operandi
- il quoziente della divisione viene normalizzato, cioè l'esponente è aggiustato in modo che il bit più significativo della mantissa sia 1.
- se il quoziente risulta essere in un registro più lungo del formato massimo consentito della virgola mobile, grazie all'utilizzo dei formati estesi per i risultati intermedi, è necessario arrotondarlo. Tale arrotondamento può avvenire per: arrotondamento al più vicino, per eccesso, per troncamento.

2. **Descrivere i possibili approcci per trattare l'indirizzo di ritorno da una chiamata di procedura.**

Soluzione
L'indirizzo di ritorno da una chiamata di procedura permette all'istruzione di Return della procedura di tornare nell'appropriata sezione di codice, cioè in quella dove era avvenuto il Call.
Tale indirizzo può essere memorizzato in un registro, all'inizio della procedura chiamata o in cima alla pila.
La memorizzazione nel registro prevede l'utilizzo di un registro specializzato dove verrà salvato l'indirizzo dell'istruzione successiva a quella che ha fatto la chiamata. Verrà poi caricato nel PC l'indirizzo della prima istruzione della procedura chiamata, che al return andrà a leggere il registro specializzato.
Detti: RN = registro specializzato, Δ = lunghezza di una istruzione, PC = program counter, X = indirizzo della procedura chiamata. Volendo schematizzare:

- $RN = PC + \Delta$
- $PC = X$

La memorizzazione all'inizio della procedura chiamata salva l'indirizzo dell'istruzione successiva a quella che ha chiamato la proceduta all'inizio della procedura chiamata. Viene poi copiato sul PC l'indirizzo della seconda istruzione della procedura, contenendo la prima l'indirizzo di ritono e la seconda la vera istruzione da eseguire. Volendo schematizzare:

- $X = PC + \Delta$
- $PC = X + 1$

Questi due metodi di memorizzazione non permettono la gestione di chiamate annidate, la quale può avvenire solo mediante la memorizzazione in cima alla pila.

Con la memorizzazione in cima alla pila gli indirizzi di ritorno vengono memorizzati uno dopo l'altro in cima alla pila, e vengono presi nell'ordine inverso all'inserimento alla chiusura delle procedure. La pila è una porzione di memoria riservata dove le scritture e le letture avvengono sempre in cima. Il top della pila è indicizzato da un apposito registro della CPU, lo stack pointer SP. Questo tipo di memorizzazione permette la gestione di chiamate annidate. Ad esempio avendo il Call della procedura 1 l'indirizzo di ritorno di 1 viene registrato sul top della pila, ad un Call annidato di una procedura 2 l'indirizzo di ritorno di 2 viene registrato sul top della pila ed l'indirizzo di ritorno di 1 va in seconda posizione. Quando 2 farà il return sulla pila leggerà il corretto indirizzo, cancellandolo, a questo seguirà il return di 1 la quale troverà ancora il corretto indirizzo, cancellandolo.

3. **Nella pipeline MIPS, spiegare come lo stadio ID è in grado di rilevare la dipendenza dei dati.**

Soluzione - slide+registrazioni... no libro =(
Lo stadio ID è in grado di rilevare la dipendenza dei dati grazie ad un apposito circuito di identificazione delle dipendenze e gestione del dataforwarding. Tale circuito ha tre componenti fondamentali:

- Forwarding Unit: unità che decide se attivare il forward attivando nell'opportuno modo i multiplexer della ALU
- Hazard detection Unit: unità in grado di riconoscere le dipendenze e di generare stalli in caso di dipendenze non risolvibili
- Control Unit: manda segnali di controllo che regolano il forward ed i dati che devono essere mandati (inoltre manda segnali di controllo che regolano l'esecuzione e l'utilizzo dell'hardware per la memorizzazione ed il write back)

La Forwarding Unit è in grado di rilevare le dipendenze confrontando, mediante comparatori, i registri associati ai campi di input dell'istruzione che si trovano nello stadio ID con i registri target delle istruzioni precedente non ancora terminate. I comparatori possono leggere i registri associati ai campi di input e di output facendo riferimento al campo IR dei banchi di registri. L'istruzione che è nello stadio ID avrà le informazioni sui/sul registri/registro di input sempre nel banco IF/ID, mentre il registro dell'istruzione con cui fare il confronto sarà indicato nel banco ID/EX o EX/MEM o MEM/WB.
Se i due registri sono uguali e non vi è possibilità di dataforwarding viene generato uno stallo, altrimenti procede.
In particolare se:

- istruzione di input ha formato R i campi di input sono rs (IF/ID.IR[rs]) ed rt (IF/ID.IR[rt]) e verranno confrontati con:
    - rd se l'istruzione precedente ha formato R
    - rt se l'istruzione precedente ha formato I
- istruzione di input ha formato I il campo di input è rs (IF/ID.IR[rs]) e verrà confrontato con:
    - rd se l'istruzione precedente ha formato R
    - rt se l'istruzione precedente ha formato I

4. **Spiegare le motivazioni alla base dell'architettura CISC.**

Soluzione
L'architettura CISC nacque per diversi motivi, di cui i più importanti sono:

- facilitare la scrittura del compilatore
- supportare i linguaggi ad alto livello sempre più complessi

- migliorare l'efficienza dell'esecuzione, in quanto essendo state implementate operazioni più complesse tramite microcodice, si dovrebbe aver avuto un incremento prestazionale non dovendo fare una successione di istruzioni primitive per eseguire l'istruzione complessa.
- si pensava di poter ottenere programmi più piccoli, che occupassero meno memoria e fossero eseguiti più velocemente
- facilitare il lavoro del programmatore a discapito dell'aumento del costo dell'hardware. Il costo del software è molto maggiore del costo dell'hardware, in quanto l'hardware può essere ammortizzato nel tempo, mentre vengono spese migliaia di ore uomo su uno stesso computer.
- permettere grande flessibilità mettendo a disposizione un ampio set di istruzioni e svariati metodi di indirizzamento.

by Caesar

NON HO LA PRESUZIONE DI DIRE CHE QUESTO E' IL CORRETTO MODO DI SVOLGERE IL COMPITINO.
LO CARICO PERCHE' POTREBBE ESSERE UTILE AVERE UN CONFRONTO O MAGARI UNO SPUNTO PER RISOLVERE ALCUNI ESERCIZI.
SE VEDETE ERRORI AGGIORNATE PURE IL FILE CON UNA NUOVA VERSIONE CORRETTA E MIGLIORATA.
ABBIATE SPIRITO CRITICO =)

SALUTI

Caesar