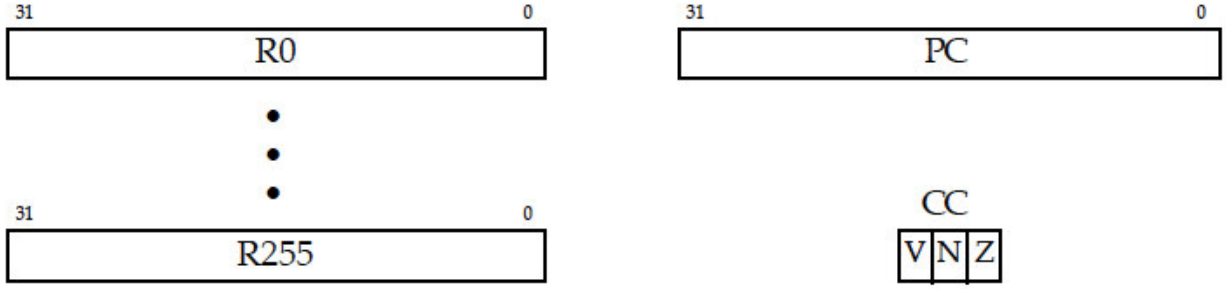
Tema n. 2

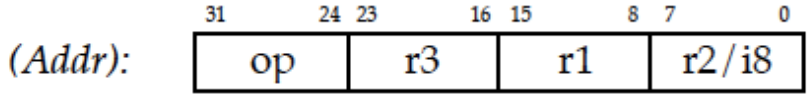
Si debba progettare un processore a 32 bit con un semplice set di istruzioni. L'organizzazione dei registri interni sia quella riportata, con 256 registri general-purpose (R0-R255), un program-counter (PC) e un registro con le condition-code (CC) in grado di memorizzare i flag interni di Overflow (V), Segno –negativo- (N) e Zero (Z) dopo ogni operazione aritmetico/logico.



Al Reset il PC è inizializzato a 0.
Le operazioni in memoria consistono in accessi a parole di 32 bit indirizzate da un indirizzo sempre su 32 bit. Le istruzioni sono memorizzate nella stessa memoria e codificate in parole da 32 bit indirizzate dal PC. Le operazioni in memoria sono sincrone e non pipelined: si accede alla memoria con l'indirizzo della locazione e si scrive/legge il dato nella cella indirizzata.
Dopo ogni lettura di una parola dell'istruzione (fetch) il PC è incrementato di uno e punta alla parola successiva.

Il set di istruzioni è codificato secondo un campo OPCODE seguito da uno o più campi per i parametri.

Le istruzioni logiche/aritmetiche e di movimentazione dei dati tra registri hanno il seguente formato:



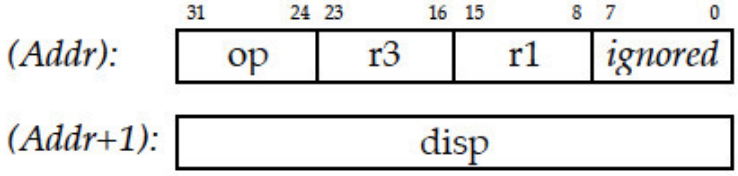
dove "op" è il campo del codice operativo, "r3" il campo del registro destinazione, "r1" il campo del primo registro sorgente e "r2/i8" il campo in cui c'è il secondo registro sorgente o un immediato (i8) su 8 bit.

Le istruzioni aritmetico/logiche e l'istruzione di trasferimento di un dato tra registri (Mov) sono riportate nella seguente tabella:

ISTRUZIONE	NOME	FUNZIONE	OPCODE
Add	Add	R3 ← R1+R2	X"00"
Sub	Subtract	R3 ← R1- R2	X"01"
Land	Logical and	R3 ← R1 and R2	X"02"
Lor	Logical or	R3 ← R1 or R2	X"03"
Lxor	Logical xor	R3 ← R1 xor R2	X"04"
Addq	Add quick	R3 ← R1+ i8	X"05"
Subq	Subtract quick	R3 ← R1 – i8	X"06"
Mov	Mov	R3 ← R1	X"07"

(X" significa che il numero che segue è in esadecimale)

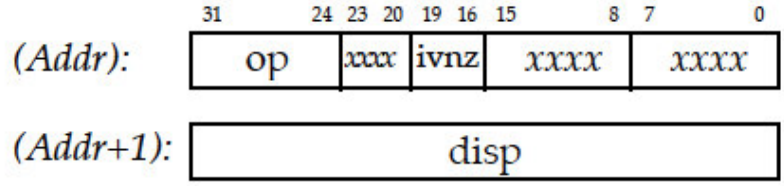
E' possibile leggere o scrivere dei dati in memoria attraverso due istruzioni (Ld e St) che permettono di indirizzare come locazione di memoria il contenuto di un registro con un displacement (offset) definito su 32 bit e definito nella parola successiva del codice a quella in cui si trova il codice dell'accesso in memoria.
Il formato delle istruzioni di load e store è:



dove "r3" è il registro che contiene il dato da scrivere/leggere e "r1" è usato come index register mentre "disp" è l'offset da sommare a "r1" per ottenere l'indirizzo completo.

ISTRUZIONE	NOME	FUNZIONE	OPCODE
Ld	Load	R3 ← M [R1 + disp]	X"10"
St	store	M [R1 + disp] ← R3	X"11"

Il processore ha poi un formato di istruzione per la gestione dei salti (Br-*ivnz*) condizionati e incondizionati secondo il seguente formato:



dove OP = X"20", "disp" è l'offset da sommare al valore attuale del PC per generare l'indirizzo cui saltare e "ivnz" è la condizione di mascheratura del salto (i campi etichettati con le xxx sono non significativi e quindi da ignorare);
il salto è effettuato se è vera la condizione *cond*

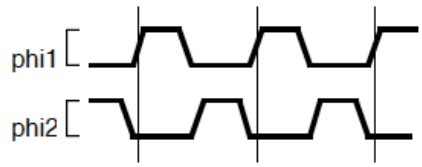
cond = [((V &v) | (N & n) | (Z & z)) = i]

dove "V", "N" e "Z" sono i flag del registro CC, mentre "i", "v", "n" e "z" sono i bit 19, 18, 17 e 16 dell'istruzione come riportato in figura ("|" e' il simbolo dell'or e "&" il simbolo logico dell'and).
Ad esempio per effettuare il salto sulla condizione Z = 1 deve essere ivnz= 1001 , mentre il salto sulla condizione V= 0 avra' ivnz = 0100. Il salto incondizionato avra' infine ivnz=0000.

Il formato dell'istruzione è quindi:

ISTRUZIONE	NOME	FUNZIONE	OPCODE
Br-*ivnz*	branch	If *cond* Then PC ← PC + disp	X"20"

Il processore lavora con un timing basato su un clock sincrono a due fasi (phi1 e phi2) non overlapped che vengono iniettate tramite due terminali di ingresso e generate all'esterno del dispositivo.



Il processore deve iniziare l'esecuzione di una nuova istruzione ad ogni colpo di clock (a parte le "Ld", le "St" e i salti che richiedendo due cicli di fetch necessitano ovviamente di due colpi di clock per completare la fase di fetch).

Si lascia ampia libertà al candidato di definire i segnali di interfaccia necessari per collegare il processore con una memoria esterna, fermo restando il vincolo che l'accesso alla memoria deve essere sincrono all'interno di un colpo di clock (ad esempio su phi1=1 vengono inviati gli indirizzi e su phi2=1 prelevati i dati).

Il candidato, sulla base delle proprie conoscenze e di quanto specificato in precedenza, sviluppi i seguenti punti:

1. Studio dell'organizzazione interna del processore al fine di eseguire un flusso di istruzioni predefinito nella memoria esterna dove sono memorizzati anche i dati non salvati nei registri interni. Si considerino eventuali criticità e si suggeriscano eventuali modifiche architetturali per rendere il progetto più efficiente.
2. Studio e definizione del protocollo di accesso e delle tempistiche della memoria esterna al fine di ottimizzare l'interfaccia memoria/processore
3. Progetto e descrizione in VHDL sintetizzabile del processore corredato da un'analisi critica delle prestazioni ottenibili con le tecnologie attuali
4. Definizione di un Test-bench (in VHDL) in grado di validare il progetto del processore

Tema n. 3

Nell'ambito dei dispositivi MEMS (Micro Electro Mechanical System) rivestono particolare interesse le strutture "Lab-on-Chip" impiegate per analisi di tipo biologica e chimica.
Il Candidato, dopo aver definito un settore aplicativo biomedicale, progetti una struttura Lab-on-Chip.
In particolare:

1. si indichino i principi di funzionamento del dispositivo;
2. si riporti uno schematico del dispositivo;
3. si descriva il flusso dei passi di processo;
4. si dettagli uno dei passi di processo proposti.

Viene richiesto inoltre di affrontare il problema del packaging del dispositivo e l'integrazione dell'elettronica di controllo.
In fase di valutazione sarà apprezzata una trattazione con scelte motivate, precise e schematiche.