

Laboratorio di Sistemi per la Progettazione Automatica

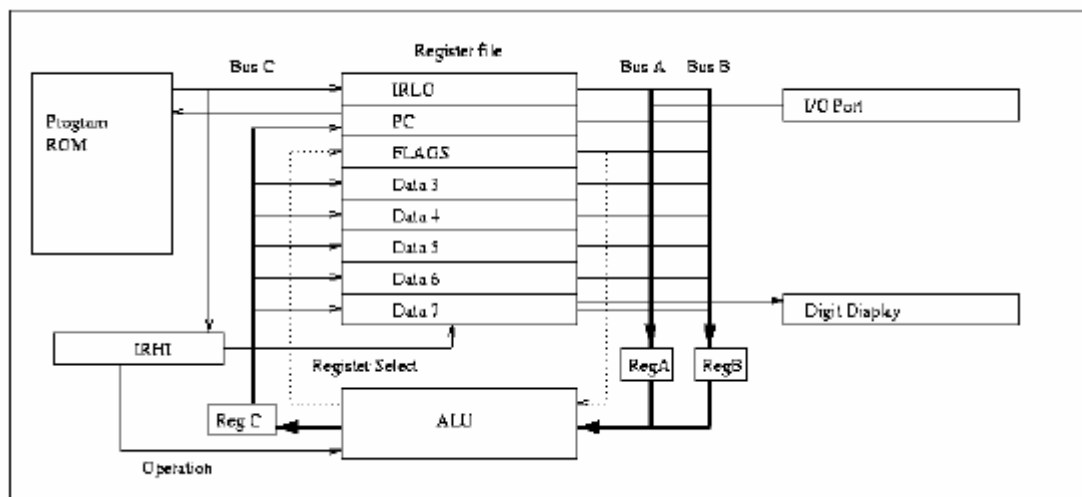
A.A. 2006/2007

Nicola Bombieri
Università degli Studi di Verona
Dipartimento di Informatica

Elaborato finale

Si modelli in VHDL la CPU riportata nello schema come una macchina a stati finiti (4 stati). L'elaborato sarà considerato completo e potrà essere consegnato quando saranno stati portati a termine i seguenti punti:

1. modello VHDL in stile behavioral della CPU e sua simulazione
2. versione sintetizzata del modello al punto 1 e confronto simulazioni
3. relazione conclusiva (4-5 pagine in formato pdf)



Caratteristiche della CPU

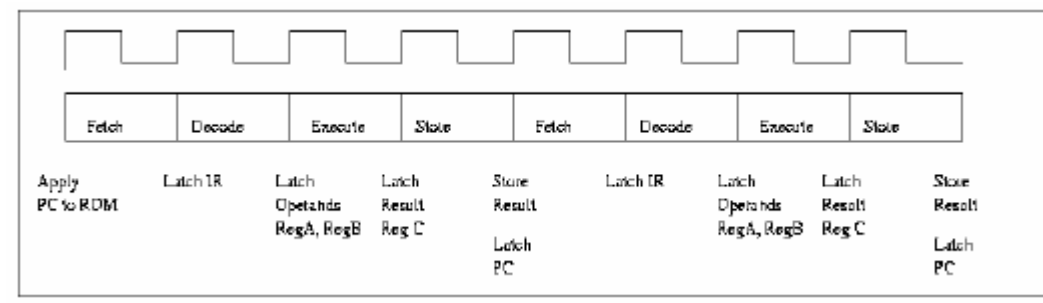
L'istruzione set della CPU da realizzare è composto da 16 istruzioni:

1. (00000) LOAD {Register A} : Loads the specified register with the value in the data field of the instruction
2. (00001) MOVE {Register A} {Register B}: Moves the contents of register B to register A
3. (00010) BRA {Address}: Branches to the specified address.
4. (00011) COMP {Register A}: Takes the bit-wise complement of the specified register and puts the result back there.
5. (00100) INC {Register A}: Increments the value in register A and puts the result back there.
6. (00101) SLL {Register A}: Shift left logical.
7. (00110) SRL {Register A}: Shift Right logical.
8. (00111) SRA {Register A}: Shift Right Arithmetic.

9. (01000) BRZA {address}: Branches to the specified address if the last result was zero.
10. (01001) BRZ {offset}: Branches to the current address plus the specified offset if the last result was zero.
11. (01010) BRN {offset}: Branches to the current address plus the specified offset if the last results was negative.
12. (01011) ADD {Register A} {Register B}: Adds the two registers together and puts the result in register A.
13. (01100) DEC {Register A}: Adds -1 to register A and puts the result in register A.
14. (01101) AND {Register A} {Register B}: ANDs the two registers together and puts the result in register A.
15. (01110) OR {Register A} {Register B}: ORs the two registers together and puts the result in register A.
16. (01111) LOADIO {Register A} {IO Location}: Loads an 8-bit value from an external source (e.g. DIP switches) and puts the result in register A.

Ogni istruzione è definita da 20 bit secondo il seguente schema:

- bit 19 : non utilizzato
- bit 18-14: codice operative istruzione
- bit 13-11: indice registro utilizzato per l'input 1 della ALU e del registro dove verrà memorizzato il risultato della ALU
- bit 10- 8 : indice registro utilizzato per l'input 2 della ALU
- bit 7- 0 : il valore a 8 bit del valore caricato dall'istruzione LOAD o l'indirizzo/spiazzamento dell'istruzione BRANCH



ALU:

L'unità aritmetico logica seleziona l'operazione da eseguire mediante un std_logic_vector a 4 bit (porta OP). Tale operazione viene eseguita sui 2 operandi in ingresso di tipo unsigned ad 8 bit (porte IN1 e IN2). Il risultato della ALU è un unsigned ad 8 bit (OUT), essa è inoltre in grado di impostare un'uscita std_logic_vector a 2 bit rappresentante una flag (porta FLAG) secondo il seguente significato:

1. FLAG(0) = '1' se l'ultima operazione eseguita ha prodotto il risultato 0
2. FLAG(1) = '1' se l'ultima operazione eseguita ha prodotto un risultato < 0

Le operazioni che l'ALU deve poter eseguire sono le seguenti:

1. Selector input = "0000", "0001": Pass input 2 through the ALU, setting the flags.
2. Selector input = "0010", "1000": Pass input 2 through the ALU, but don't change the flags.
3. Selector input = "0011": Bit-invert input 1.
4. Selector input = "0100": Increment input 1.
5. Selector input = "0101": Shift input 1 left by one position and fill with a '0'.
6. Selector input = "0110": Shift input 1 right by one position and fill with a '0'.
7. Selector input = "0111": Shift input 1 right while maintaining the sign of input 1.
8. Selector input = "1001", "1010": Add input 1 to input2 without setting the flags.
9. Selector input = "1011": Add input 1 to input 2.
10. Selector input = "1100": Add -1 (0xFF) to input1.
11. Selector input = "1101": AND input 1 with input2.
12. Selector input = "1110": OR input 1 with input 2.
13. Selector input = "1111": Pass input 2 through the ALU, setting the flags.

L'uscita FLAG deve essere impostata ad ogni operazione, se non diversamente specificato.

Registri:

L'architettura deve contenere 3 registri di controllo (IR, PC, FLAG) e 5 registri dati (D3,...D7). I registri dati possono essere definiti come un array di 5 elementi da 8 bit unsigned indicizzati da 3 a 7. Questi indici corrisponderanno ai 3 bit del codice operativo delle istruzioni. I codici dei registri di controllo sono:

1. [000] per IRLO e conterrà gli 8 bit meno significativi del codice istruzione in esecuzione
2. [001] per il program counter PC
3. [010] per il registro FLAG del quale solo i 2 bit meno significativi saranno utilizzati

Dovrà essere dichiarato anche il registro IRHI, che conterrà i 10 bit dell'operazione riguardanti: codice operazione ALU (4 bit), registro di input 1 dell'ALU (3 bit) e registro di input 2 dell'ALU (3 bit).

Dichiarazione Macchina a Stati:

Si crei un processo sensibile ai segnali di reset e di clock (attivi alti), composto da 4 stati Fetch (F), Decode (D), Execute (E) e Store (S). Il reset sincrono azzerà tutti i registri, compreso il PC e riporta il micro nello stato di Fetch.

- Nello stato F il contenuto del PC viene passato ad una ROM che restituisce la prossima istruzione da eseguire. Il codice istruzione deve essere memorizzato nei registri IRLO e IRHI al prossimo fronte di salita del clock e il processo deve avanzare allo stato D.
- Nello stato D, l'istruzione memorizzata in IR deve essere decodificata e i due operandi dell'ALU sono memorizzati in due registri temporanei al prossimo fronte di salita del clock. Al fronte di salita del clock successivo il processo deve avanzare allo stato E.
- Nello stato E gli input specificati dall'istruzione devono essere passati all'ALU unitamente al codice operazione, come specificato in IRHI. Al seguente fronte di salita del clock, il

risultato dell'ALU deve essere memorizzato in un registro temporaneo e il processo deve avanzare allo stato S.

- Nello stato S, il contenuto del registro temporaneo deve essere copiato nell'apposita locazione (si utilizzi uno statement case per discriminare tra il diverso comportamento dell'istruzione eseguita). Il PC deve essere incrementato per eseguire la prossima istruzione, se l'istruzione eseguita non è stata un salto incondizionato. Nota: il PC non deve essere incrementato utilizzando la ALU. Le azioni legate a questo stato si concludono riportando il processo allo stadio F.

Come testare la CPU:

Scrivere un programma, per la CPU sviluppata, che calcoli il fattoriale di un naturale n e ne memorizzi il risultato nel registro D7.