1 Esercizi con architettura a 1 bus

1.1 Fetch dell'istruzione

NOTA: l'istruzione $ClearY$ azzera il registro Y mentre l'istruzione CB imposta a 1 il bit di riporto/prestito in modo da sommare/sottrarre 1.

1. $PC_{out}, MAR_{in}, READ, ClearY, CB, ADD, Z_{in}$
2. $Z_{out}, PC_{in}, WMFC$
3. MDR_{out}, IR_{in}

1.2 INC %eax

NOTA: per ogni istruzione, mostriamo solo la parte di microprogramma che segue il prelievo dell'istruzione.

4. $EAX_{out}, ClearY, CB, ADD, Z_{in}$
5. Z_{out}, EAX_{in}, END

1.3 INC variabile

NOTA: si assume che la variabile sia costituita da un indirizzo immediato direttamente codificato dentro l'istruzione ($addr\_field\_IR$).

4. $addr\_field\_IR_{out}, MAR_{in}, READ$
5. $WMFC$
6. $MDR_{out}, ClearY, CB, ADD, Z_{in}$
7. $Z_{out}, MDR_{in}, WRITE$
8. $WMFC$
9. END

1.4 JECXZ etichetta

NOTE:

- si assume che l'etichetta sia costituita da un indirizzo immediato direttamente codificato dentro l'istruzione ($addr\_field\_IR$);
- salvo diversamente specificato, si assume che il salto sia *relativo* in quanto l'indirizzo di salto e' specificato tramite un'etichetta simbolica.

4. $ECX_{out}, ClearY, ADD$
5. $PC_{out}, Y_{in}, if(Z_{flag} = 0) then END$
6. $addr\_field\_IR_{out}, ADD, Z_{in}$
7. Z_{out}, PC_{in}, END

1.5 PUSH %eax

NOTA: si assume l'utilizzo di una architettura Intel dove lo stack cresce verso indirizzi di memoria inferiori e %ESP punta alla cima occupata dello stack.

4. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
5. Z_{out}, ESP_{in}
6. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
7. Z_{out}, ESP_{in}
8. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
9. Z_{out}, ESP_{in}
10. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
11. $Z_{out}, MAR_{in}, ESP_{in}$
12. $EAX_{out}, MDR_{in}, WRITE$
13. $WMFC$
14. END

1.6 POP %eax

NOTA: si assume l'utilizzo di una architettura Intel dove lo stack cresce verso indirizzi di memoria inferiori e %ESP punta alla cima occupata dello stack.

4. $ESP_{out}, CB, ClearY, ADD, Z_{in}, MAR_{in}, READ$
5. Z_{out}, ESP_{in}
6. $ESP_{out}, CB, ClearY, ADD, Z_{in}$
7. Z_{out}, ESP_{in}
8. $ESP_{out}, CB, ClearY, ADD, Z_{in}$
9. Z_{out}, ESP_{in}
10. $ESP_{out}, CB, ClearY, ADD, Z_{in}$
11. $Z_{out}, ESP_{in}, WMFC$
12. MDR_{out}, EAX_{in}, END

1.7 CALL etichetta

NOTE:

- si assume che l'etichetta sia costituita da un indirizzo immediato direttamente codificato dentro l'istruzione ($addr\_field\_IR$);
- salvo diversamente specificato, si assume che il salto sia *relativo* in quanto l'indirizzo di salto e' specificato tramite un'etichetta simbolica;
- si assume l'utilizzo di una architettura Intel dove lo stack cresce verso indirizzi di memoria inferiori e %ESP punta alla cima occupata dello stack.

4. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
5. Z_{out}, ESP_{in}

6. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
7. Z_{out}, ESP_{in}
8. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
9. Z_{out}, ESP_{in}
10. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
11. $Z_{out}, MAR_{in}, ESP_{in}$
12. $PC_{out}, MDR_{in}, WRITE, Y_{in}$
13. $addr\_field\_IR_{out}, ADD, Z_{in}, WMFC$
14. Z_{out}, PC_{in}, END

1.8 CALL (%eax, %ebx)

NOTE:

- si assume che l'indirizzo a cui saltare sia memorizzato nella locazione di memoria puntata dal valore %eax+%ebx (indirizzamento indiretto);
- salvo diversamente specificato, si assume che il salto sia *assoluto*;
- si assume l'utilizzo di una architettura Intel dove lo stack cresce verso indirizzi di memoria inferiori e %ESP punta alla cima occupata dello stack.

4. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
5. Z_{out}, ESP_{in}
6. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
7. Z_{out}, ESP_{in}
8. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
9. Z_{out}, ESP_{in}
10. $ESP_{out}, CB, ClearY, SUB, Z_{in}$
11. $Z_{out}, MAR_{in}, ESP_{in}$
12. $PC_{out}, MDR_{in}, WRITE$
13. EAX_{out}, Y_{in}
14. $EBX_{out}, ADD, Z_{in}, WMFC$
15. $Z_{out}, MAR_{in}, READ$
16. $WMFC$
17. MDR_{out}, PC_{in}, END

1.9 RET

NOTA: si assume l'utilizzo di una architettura Intel dove lo stack cresce verso indirizzi di memoria inferiori e %ESP punta alla cima occupata dello stack.

4. $ESP_{out}, CB, ClearY, ADD, Z_{in}, MAR_{in}, READ$
5. Z_{out}, ESP_{in}
6. $ESP_{out}, CB, ClearY, ADD, Z_{in}$
7. Z_{out}, ESP_{in}
8. $ESP_{out}, CB, ClearY, ADD, Z_{in}$

9. Z_{out}, ESP_{in}
10. $ESP_{out}, CB, ClearY, ADD, Z_{in}$
11. $Z_{out}, ESP_{in}, WMFC$
12. MDR_{out}, PC_{in}, END

1.10 XCHG variabile, %eax

NOTA: si assume che la variabile sia costituita da un indirizzo immediato direttamente codificato dentro l'istruzione ($addr\_field\_IR$).

4. $addr\_field\_IR_{out}, MAR_{in}, READ$
5. $EAX_{out}, Y_{in}, WMFC$
6. MDR_{out}, EAX_{in}
7. $Y_{out}, MDR_{in}, WRITE$
8. $WMFC$
9. END

1.11 Considerazioni finali

Come si puo' notare contando il numero di righe di microprogramma, le istruzioni che fanno accesso alla memoria impiegano piu' cicli di clock di quelle che fanno accesso solo ai registri della CPU; a questo bisogna aggiungere che il tempo di accesso alla memoria e' maggiore di quello ai registri della CPU. Quindi sarebbe opportuno che le CPU avessero un buon numero di registri interni.

2 Esercizi con architettura a 3 bus

NOTE:

- Per lo schema della CPU fare riferimento alla Figura 3.13 di pag. 137 dell'Hamacher.
- Il comando NOP dell'ALU indica che il contenuto del bus A viene ricopiato sul bus C.
- Il comando $ClearB$ mette a zero l'ingresso B dell'ALU.

2.1 Fetch dell'istruzione

1. $PC_{outA}, NOP, MAR_{in}, READ$
2. $PC_{outA}, SetCarryIn, ClearB, ADD, PC_{in}, WMFC$
3. MDR_{outB}, IR_{in}, END

2.2 INC %eax

4. $EAX_{outA}, SetCarryIn, ClearB, ADD, EAX_{in}, END$

2.3 INC variabile

NOTA: si assume che la variabile sia costituita da un indirizzo immediato direttamente codificato dentro l'istruzione ($addr\_field\_IR$) e che l'instruction register sia connesso in uscita al bus C.

4. $addr\_field\_IR_{outC}, MAR_{in}, READ$
5. $WMFC$
6. $MDR_{outA}, SetCarryIn, ClearB, ADD, MDR_{in}, WRITE$
7. $WMFC$
8. END

2.4 XCHG variabile, %eax

NOTA: si assume che la variabile sia costituita da un indirizzo immediato direttamente codificato dentro l'istruzione ($addr\_field\_IR$) e che l'instruction register sia connesso in uscita al bus C.

4. $addr\_field\_IR_{outC}, MAR_{in}, READ$
5. $EAX_{outA}, NOP, Temp_{in}, WMFC$
6. $MDR_{outA}, NOP, EAX_{in}$
7. $Temp_{outA}, NOP, MDR_{in}, WRITE$
8. $WMFC$
9. END

2.5 Considerazioni finali

- Nell'architettura a 3 bus tutti i registri devono essere realizzati con flip-flop master/slave perche' devono essere letti e scritti contemporaneamente.
- Confrontando la lunghezza delle istruzioni nelle due architetture si vede che le uniche istruzioni che traggono giovamento dall'architettura a 3 bus sono quelle che operano solo su registri interni alla CPU; l'adozione di una architettura a 3 bus quindi ha senso solo in unione alla presenza di istruzioni semplici che non accedono alla memoria principale e alla disponibilita' di molti registri interni alla CPU (caratteristiche delle architetture RISC come quella del PowerPC).