La Control Unit genera le micro-operazioni la cui sequenza equivale a vere istruzioni Assembly. Proviamo adesso a capire come le genera. Immaginiamoci di avere queste due istruzioni nel programma in esecuzione

Istruzione 1: JNZ (%EAX) relativo

1. $PC_{OUT}, MAR_{IN}, READ, SELECT_4, ADD, Z_{IN}$
2. $WMFC, Z_{OUT}, PC_{IN}$
3. MDR_{OUT}, IR_{IN}
4. $(ZERO{==}1\ END), EAX_{OUT}, MAR_{IN}, READ$
5. $WMFC, PC_{OUT}, Y_{IN}$
6. $MDR_{OUT}, SELECT_Y, ADD, Z_{IN}$
7. Z_{OUT}, PC_{IN}, END

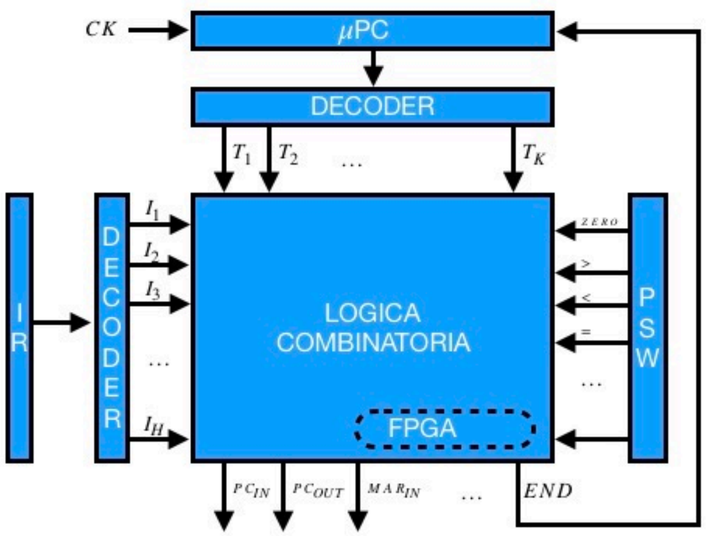
Istruzione 2: MOVL $8, (%ECX)

1. $PC_{OUT}, MAR_{IN}, READ, SELECT_4, ADD, Z_{IN}$
2. $WMFC, Z_{OUT}, PC_{IN}$
3. MDR_{OUT}, IR_{IN}
4. $OFFSET(IR)_{OUT}, MDR_{IN}$
5. $ECX_{OUT}, MAR_{IN}, WRITE$
6. $WMFC$
7. END

Ogni istruzione è identificata da un numero ($I_1, I_2, \ldots, I_H$) e per ogni istruzione abbiamo una sequenza specifica di micro-operazioni ($T_1, T_2, \ldots, T_K$). L'obiettivo è di trovare una espressione booleana per ogni segnale di uscita della CU ($PC_{IN}, PC_{OUT}, MAR_{IN}, ecc$). Come si fa? Proviamo con PC_{IN}. Cerchiamo tutte le sue ricorrenze in tutte le micro-operazioni di tutte le istruzioni. Cioè sappiamo che compare al tempo 2 e 7 in I_1 e al tempo 2 in I_2. Il ragionamento è simile per la END

$$PC_{IN} = T_2 I_1 + T_7 I_1 + T_2 I_2$$
$$END = T_7 I_1 + T_7 I_2 + T_4 I_1 ZERO$$

A livello circuitale abbiamo con contatore (Micro Program Counter) che abilita in ordine grazie a un decoder i segnali $T_1, T_2, \ldots, T_K$. Poi con lo stesso meccanismo l'istruzione è convertita in un numero $I_1, I_2, \ldots, I_H$. Nota che il segnale di RESET del contatore è proprio il segnale END



Il modello appena esposto è chiamato CPU con unità di controllo cablata, cioè la ISA del microprocessore può essere una e una sola, senza mai riuscire a cambiarla se non intervenendo proprio sulla logica combinatoria. Ma è davvero necessario cambiare l'ISA?

Supponiamo di voler riconoscere un volto. Per farlo vengono usati degli algoritmi fra cui c'è il calcolo della distanza fra due punti localizzati sulla faccia. Se queste distanze sono uguali allora lo sono anche i due volti (quello memorizzato e quello rilevato). In C questo algoritmo avrà un certo numero di istruzioni che per ipotesi corrispondono a circa 50 istruzioni Assembly e se facciamo finta di sapere come ottenere un CPI = 1 ogni calcolo richiede 50 cicli di clock. Se trovassimo un modo di eseguire lo stesso calcolo con una sola istruzione Assembly avremo risparmiato 50 cicli, ovvero l'algoritmo potrebbe essere 50 volte più veloce (se dipende molto da questo calcolo). Grazie a questa idea hanno pensato di creare parte della CU con un FPGA, permettendo così di avere una porzione di ISA configurabile a piacere. Questo è tra l'altro lo stato d'arte attuale

Prima di questo le CPU esistevano ma non erano cablate. Infatti mancavano gli strumenti di sintesi (come SIS) necessari a progettare circuiti così complessi. Usavano un'altra tattica che per certi versi era molto più potente. Possiamo osservare che ad ogni ciclo di clock l'uscita della CU è semplicemente una parola costituita da zeri e uni. Nulla vieta di salvare queste parole in sequenza in una memoria e richiamarle una alla volta, si chiamavano CPU micro-programmate. Pensandoci questo metodo è davvero interessante perché l'intera ISA fa parte del programma. Riusciresti a cambiare ISA sostituendo il programma. Eppure nessuno fa più così, perché? Beh, un accesso alla memoria non è di certo la cosa più veloce del mondo e se per ogni singola micro-operazione serve accedere alla memoria le performance sono davvero preoccupanti. In ogni caso oggi è possibile richiamare quell'idea e portare memorie molto vicine al processore così da mantenere un certo grado di flessibilità assieme a prestazioni soddisfacenti