Transactor Lessons

Walter Cazzola

COLLABORATORS

	TITLE : Transactor Lessons	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Walter Cazzola	February 14, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Transactor Lessons **1**

1.1 Assembler Prima Lezione 1

Chapter 1

Transactor Lessons

1.1 Assembler Prima Lezione

Prima Lezione: Conoscere la CPU

Il computer funziona con l'elettricità, più precisamente i dispositivi sulla scheda madre sfruttano stati di tensione alta e bassa. Da un punto di vista logico possiamo immaginare la tensione alta come un 1 e la tensione bassa come uno 0; questo (1 o 0) è il più piccolo dato su cui si possa operare ed è detto bit. Quindi tutti i dispositivi all'interno della macchina dialogano tra loro (ad es. CPU e RAM) tramite sequenze di 1 e 0, producendone milioni al secondo. Questo è il Linguaggio Macchina. Se una stringa, composta da 1 e/o 0, è lunga 8 bit è detta Byte, se lunga 16 bit è una Word e se lunga 32 bit è una Longword. Alcune stringhe rappresentano comandi da eseguire per la CPU, altre sono dati. Ora scrivere un programma interamente in questa maniera significherebbe scrivere sequenze di 1 e 0 lunghe qualche miliardo di cifre: immaginate che stress! Oltretutto un errore sarebbe molto più complicato da rintracciare, anche perchè a questo livello non esistono errori di sintassi (ci sono solo 0 e 1!) ma solamente di tipo logico. è per questo che le stringhe che rappresentano comandi vengono classificate usando delle sigle. Ad ogni sigla corrisponde esattamente una precisa sequenza di 1 e 0 che farà eseguire qualcosa alla CPU. Questo è l'Assembler, un linguaggio che ha la stessa identica potenza del Linguaggio Macchina con in più la facilità per l'utente di poter tranquillamente ricordarsi i nomi dei comandi: per esempio, una cosa è ricordarsi la sigla ADD e un'altra è ricordarsi l'equivalente in LM che potrebbe essere 0100101011000111 (se pensiamo che il 68000 ha un set di 56 istruzioni...).

Sottolineamo un particolare: un'istruzione in Asm (o LM) che il programmatore impartisce alla macchina è la più piccola che la CPU possa svolgere (a meno che non sia microprogrammata, ma questo è un altro discorso). Ciò non è vero per i linguaggi più distaccati dalla macchina (Pascal, C...): un'istruzione `write' in questi linguaggi corrisponde ad una serie di istruzioni in Asm; un'istruzione in Asm invece va ad attivare fisicamente nella CPU l'organo preposto ad eseguire quella particolare istruzione. È per questo che l'Asm viene detto a basso livello: si va realmente a lavorare con l'hardware della CPU! Più giù di così non si può!

Dato che programmando in Assembler si programma la CPU e nient'altro, iniziamo a descrivere il 68000. Anzitutto chiariamo il concetto di registro: un registro non è altro che una cella di memoria che anzichè trovarsi fisicamente nella RAM si trova all'interno della CPU. L'uso di registri nella CPU, oltre a garantire un'elevata velocità, è di importanza fondamentale: alcuni sono vitali per lo stesso funzionamento della macchina, altri di uso generale hanno lo scopo di immagazzinare momentaneamente dati. Vediamo come sono organizzati nel 68000, descrivendo prima i registri di uso generale:

- D0, D1, D2, D3, D4, D5, D6, D7: sono 8 registri cosiddetti di dati, di

dimensione 32 bit l'uno. In ognuno di questi si può mettere il risultato

di un'operazione, dati temporanei che mi interessa salvare e qualunque

altra cosa passi per la testa. La scelta di quale registro dati usare è

completamente a discrezione del programmatore.

- A0, A1, A2, A3, A4, A5, A6, A7: sono 8 registri di indirizzo (Address),

di 32 bit l'uno, in cui memorizzare indirizzi di memoria che ci sono utili.

Vedremo che tornano molto comodi come indici quando si vogliono esaminare

dati memorizzati sequenzialmente in memoria o quando si vuole creare nuovi
stack. Il registro A7 è particolare in quanto adibito a Stack Pointer,
spiegato più sotto tra i registri fondamentali, per cui non è sfruttabile
come gli altri 7 registri indirizzo. Come per i registri dati si è liberi
di usare un qualunque registro (tranne A7).

Nessuno vieta di usare i registri dati per contenere indirizzi e i registri indirizzo per contenere dati, ma come vedremo è molto più comodo usarli secondo le idee del costruttore! Vediamo ora i registri 'vitalì:

- PC, Program Counter: questo registro contiene (in gergo, punta)
l'indirizzo della Ram in cui si trova l'istruzione da eseguire, ed è
composto da 24 bit (quindi il suo valore può variare da 0 a 16777215).

Poichè le istruzioni del 68000 sono lunghe 16 bit (words) il PC conterrà sempre un indirizzo pari (altrimenti...GURU!). Viene aggiornato automaticamente ogni volta che l'istruzione viene eseguita. L'unico modo che l'utente ha di modificare questo registro con un valore desiderato è tramite un'istruzione di salto.

- SR, Status Register: questo registro contiene i cosiddetti 'Flags', o
bandierine di segnalazione ed è composto da 16 bit, divisi in 1 byte
utente (il byte basso, dal bit 0 al bit 7) e 1 byte di sistema (divisione
dovuta al fatto che il 68000 lavora in 2 modi: utente e supervisore. Ce ne
occuperemo più in là). Il byte di sistema può essere modificato solo
quando il 68000 è in modo supervisore.

Ogni volta che viene eseguita un'istruzione che opera su un dato si può avere come risultato zero, un numero negativo, un numero troppo grande e così via. Per ognuna di queste condizioni viene messo a 1 un particolare bit (ovvero flag) dello SR. I flag del byte utente, detto anche registro dei codici-condizione, sono:

Bit 0 - C (Carry): Questo flag viene messo a 1 se un'addizione produce un
riporto o una sottrazione effettua un prestito, altrimenti và a 0. Può
anche contenere il valore di un bit dopo un'operazione di spostamento
(shift) o rotazione.

Bit 1 - V (Overflow): Questo flag è significativo solo nelle operazioni tra
numeri con segno. Viene messo ad 1 se il risultato dell'addizione tra
2 numeri di segno uguale o della sottrazione tra 2 numeri di segno
opposto supera il campo di complemento a 2 dell'operando (questa è una
questione matematica abbastanza semplice, se qualcuno non ne è a
conoscenza la spiegherò la prossima volta), altrimenti è azzerato. Può
essere settato a 1 anche quando il bit più significativo (ad es. se
operiamo su 1 byte si tratta dell'ottavo bit!) di un operando viene
modificato dopo un'istruzione SHIFT, altrimenti è azzerato.

Bit 2 - Z (Zero): Viene posto a 1 se il risultato di un'operazione è 0, facile no?

Bit 3 - N (Negativo): Ancora, è significativo solo nelle operazione tra numeri
con segno. Viene messo a 1 se il risultato di un'operazione
logico-aritmetica, di shift o di rotazione è negativo.

Bit 4 - X (Extend): Si usa come bit di carry nelle operazioni in precisione

multipla (don’t worry, ci torneremo!).

I flag del byte di sistema sono: Bit da 8 a 10 (I0, I1, I2): Maschera di interrupt. Determina il livello di richiesta di interrupt. La cosa è meno complicata di quanto possa sembrare, quando discuteremo gli interrupt torneremo ampiamente su questi flag.

Bit 13 - S (Supervisore): Se a 1 il 68000 è in modo supervisore, altrimenti in modo utente.

Bit 15 - T (Trace): Di FONDAMENTALE importanza, se settato a 1 permette di eseguire

un’istruzione alla volta bloccando ogni volta il 68000 finchè lo desidera

l’utente.

Avete mai sentito parlare di pirati che tracciano i programmi? Vi state domandando qual’è l’utilità di questo registro con i suoi flags? Molta! Il 68000, come praticamente ogni CPU, è dotato di istruzioni di salto ’condizionato’, cioè salta ad un indirizzo in memoria se si verifica una condizione, proprio una di quelle segnalate dai bit di flag dello SR. Come avrete notato non sono stati usati tutti i 16 bit dello SR: i bit inutilizzati avranno sempre valore 0. Schematicamente:

_15_____13_____10__9___8_____4___3___2___1___0_

| T |\\\| S |\\\|\\\| I2| I1| I0|\\\|\\\|\\\| X | N | Z | V | C |

- SP, Stack Pointer: Si trova in A7, contiene l’indirizzo attuale dello

stack in RAM.

Lo stack è una zona della RAM riservata per la memorizzazione temporanea di dati. Tramite una particolare istruzione si può spostare il contenuto di qualunqe registro del 68000 nello stack per poi essere ripreso quando ne avremo bisogno. La peculiarità dello stack è il funzionamento ’a catasta’: ogni volta che metto una cosa nello stack l’indirizzo nello SP cresce, così come facendo una catasta di libri per terra il primo messo sarà quello che si trova sul pavimento, il secondo sul primo e così via. Quindi il primo che riprendo sarà l’ultimo che ho messo, in cima alla catasta. Questo modo di operare è detto LIFO, Last In First Out, appunto l’ultimo dato messo nello stack è il primo a venir ripreso. Se ci si trova in modo supervisore (cioè il flag S (bit 13) dello SR è settato) A7 rappresenta l’ SSP, Puntatore di Stack Supervisore, la stessa cosa che abbiamo visto finora ma ad un indirizzo diverso (in pratica abbiamo uno stack in modo utente e uno in modo supervisore).

Bene! Questo è tutto ciò che un programmatore deve sapere (ma deve saperlo bene!) riguardo l’hardware della CPU. In futuro, a richiesta, si potrebbe vedere più approfonditamente come è fatta una CPU. Fatemi sapere!

Indice Assembler Indice Corsi

Lezione Successiva