ENIAC non era un calcolatore ma un insieme di una quarantina di moduli utilizzando i quali era possibile mettere insieme un dispositivo costruito ad hoc per ogni problema da risolvere.

Von Neumann ha raccolto alcune idee e proposte che circolavano nel gruppo ENIAC e, durante un viaggio a Los Alamos, le ha utilizzate nel predisporre un progetto organico per EDVAC che contiene i suoi contributi originali:

* il programm counter,
* un organo di controllo della istruzione corrente,
* uno insieme semplice, flessibile e (Turing) completo di istruzioni del linguaggio macchina.

EDVAC, realizzato rispettando l’architettura di von Neumann, si può considerare il primo ***computer*** della storia. Con ENIAC cambiare problema da risolvere implicava modifiche (parziali) dell’hardware; con EDSAC era sufficiente cambiare il software.

**SET ISTRUZIONI DELLA MACCHINA IAS**

**LA MACCHINA UNIVERSALE DI VON NEUMANN**

Le parole sono di 40 bit

La memoria M contiene 2000 parole M(1), M(2), … M(2000)

Nell’unità centrale ci sono due registri: MQ e AC

|  |  |
| --- | --- |
| Unità centrale | Memoria |
| Due registri MQ e AC per uno dei due operandi coinvolti e/o per il risultato | 2000 registri per dati e risultati e per le istruzioni da eseguire M(1), M(2), … M(2000) |

**Load trasferisce un dato nella unità centrale**

**Stor trasferisce un dato nella unità centrale AC nella memoria**

**CODICE OP - INDIRIZZO OPERANDO - INDIRIZZO ISTRUZIONE SUCCESSIVA**

**TRASFERIMENTO DI DATI**

**CODICE BINARIO CODICE MNEMONICO DESCRIZIONE**

00001010 LOAD MQ Transfer contents of register MQ to the

ccumulator AC

00001001 LOAD MQ,M(X) Transfer contents of memory location X

to MQ

00100001 STOR M(X) Transfer contents of accumulator to

memory location X

00000001 LOAD M(X) Transfer M(X) to the accumulator

00000010 LOAD –M(X) Transfer **- M(X)** to the accumulator

00000011 LOAD |M(X)| Transfer |M(X)| to the accumulator

00000100 LOAD –|M(X)| Transfer **-|M(X)|** to the accumulator

**TRASFERIMENTO DEL CONTROLLO**

00001101 JUMP M(X,0:19) Take next instruction from left half of M(X)

00001110 JUMP M(X,20:39) Take next instruction from right half of M(X)

00001111 JUMP+ M(X,0:19) If number in the accumulator is

nonnegative, take next instruction from

left half of M(X)

00010000 JUMP-M(X,20:39) If number in the accumulator is

no positive, take next instruction from

right half of M(X)

**ARITMETICA**

00000101 ADD M(X) Add M(X) to AC; put the result in AC

00000111 ADD |M(X)| Add |M(X)| to AC; put the result in AC

00000110 SUB M(X) Subtract M(X) from AC; put the result in AC

00001000 SUB |M(X)| Subtract |M(X)| from AC; put the

remainder in AC

00001011 MUL M(X) Multiply M(X) by MQ; put most

significant bits of result in AC, put least

significant bits in MQ

00001100 DIV M(X) Divide AC by M(X); put the quotient in

MQ and the remainder in AC

00010100 LSH Multiply accumulator by 2, i.e., shift left

one bit position

00010101 RSH Divide accumulator by 2, i.e., shift right

one position

**SOSTITUZIONE DI CONTENUTI**

00010010 STOR M(X,8:19) Replace left address field at M(X) by 12

rightmost bits of AC

00010011 STOR M(X,28:39) Replace right address field at M(X) by 12

leftmost bits of AC

--------------------------------------------------------------------------------------------------------------------