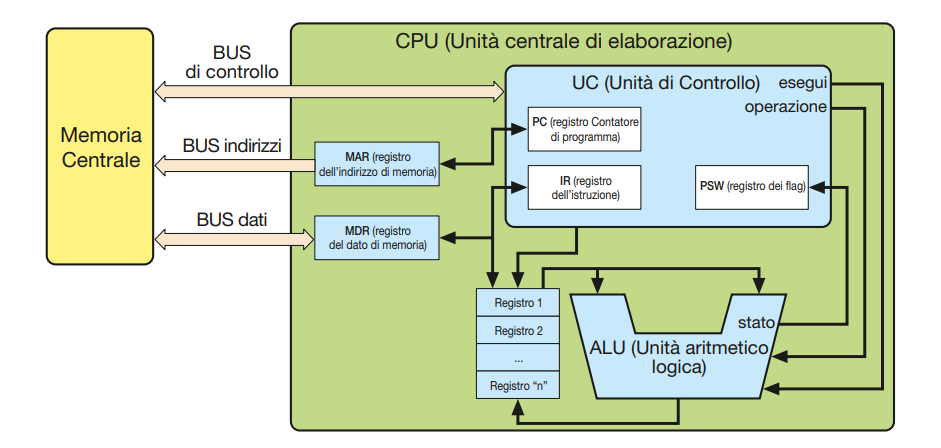
CPU (Central Processing Unit)

Il Processore è quel componente responsabile di eseguire le operazioni e regolare il traffico dati all’interno del PC.

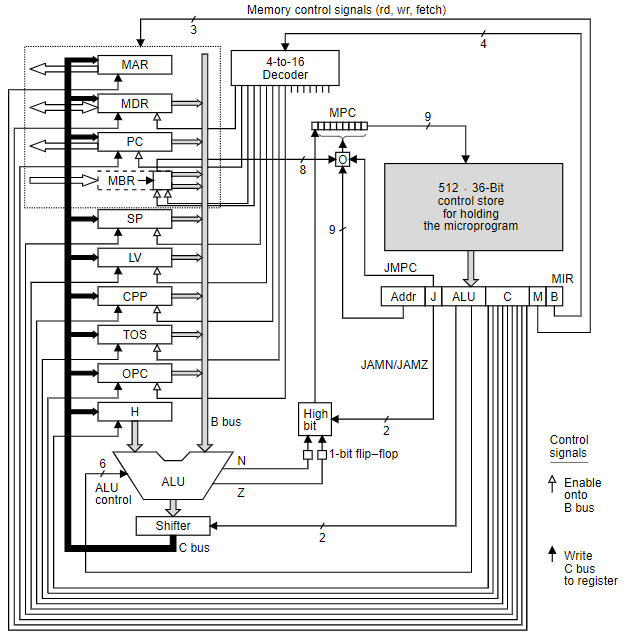
Viene definito **“General Purpose”** perché offre una grande versatilità di utilizzo.

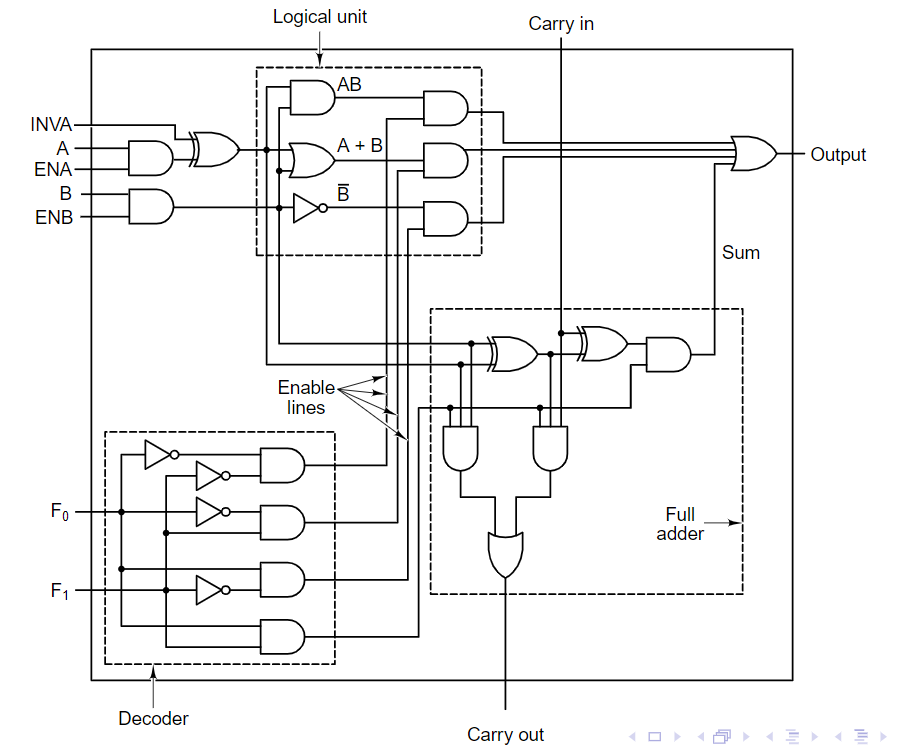
**“Special Purpose”** sono i dispositivi adatti a compiere un obiettivo ben preciso e definito, come i Router, che si occupano dell’indirizzamento.

**STRUTTURA INTERNA DELLA CPU**





* **Unità di Controllo (CU)**: preleva istruzioni e dati dalla memoria centrale (RAM o HDD/SSD). Invia i dati alla ALU e ne governa l’esecuzione. Gestisce i risultati, salvandoli in memoria. Gestisce anche gli eventuali salti di istruzione, agendo sullo stack PC, aumentando o diminuendo il IP a piacimento, in base alla prossima istruzione da eseguire
* **ALU(Ari****tmetic Logical Unit)**: ha il compito di eseguire le operazioni aritmetico logiche ai dati forniti dalla CU



F0 e F1 sono dei segnali di controllo, e attivano le linee di Enable, che attiveranno rispettivamente il circuito Logico o Aritmetico

F0 = F1 = 0, si attiva la E0 che attiva l’operazione AND

F0 = 0, F1 = 1 si attiva la E1 per l’operazione OR

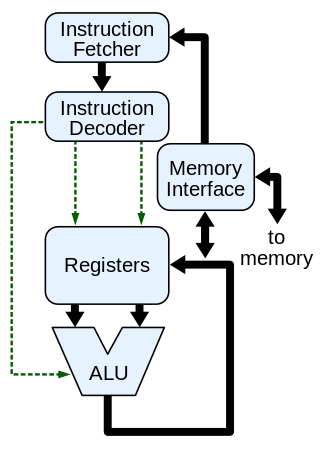
F0 = 1, F1 = 0 si attiva E2, per operazione NOT

F0 = F1 = 1 si attiva E3, che attiva il circuito aritmetico Full Adder

INVA = è un segnale di controllo che abilita o meno l’inversione di A

Carry IN = eventuale riporto dell’operazione precedente. Da carry OUT passa a carryIN

* **Shifter**: si occupa di eseguire operazioni di moltiplicazione e divisione per potenze del 2
* **Registri interni alla CPU**: hanno un tempo di accesso molto veloce, più della RAM, ma a discapito della capienza. Il valore complessivo dei dati nel registro totale identifica lo stato della CPU



* + - **REGISTRI DELLE ISTRUZIONI**
    - **PC(Program Counter) o IP(Instruction Point):** contiene l’indirizzo di memoria della prossima istruzione (linguaggio macchina) da eseguire. La CPU legge al suo interno l’indirizzo da contattare per trovare la operazione successiva. Una volta prelevato l’indirizzo il PC viene shiftato. Questo registro ha una logica LIFO (Last Input Fist Output).
      * Questo registro è un puntatore, infatti contiene l’indirizzo della cella a cui si vuole accedere. Dopo ogni prelievo il PC viene incrementato di 1, per puntare all’indirizzo successivo
    - **IR Instruction Register:** contiene l’istruzione intera prelevata e dalla memoria che andrà decodificata e poi eseguita

CICLO MACCHINA (Fetch, Decode, Execute)

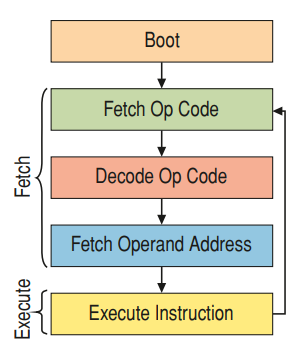
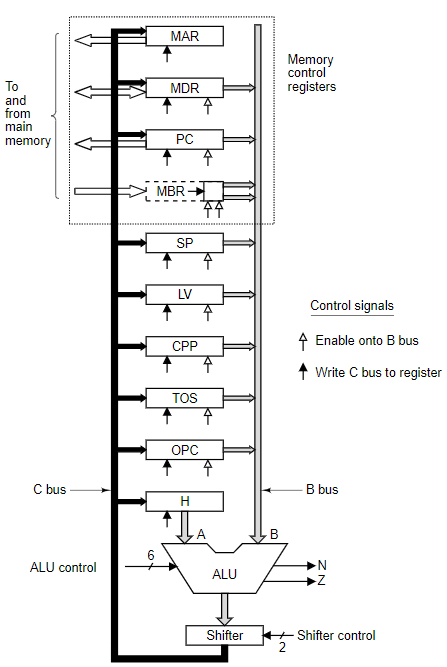


Immagine che contiene testo, elettronico

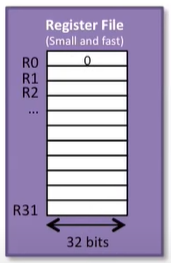
Descrizione generata automaticamente

* **Fetch dell’ISTRUZIONE:** la CPU deve comunicare con la RAM. All’interno dell’IP/PC è presente l’indirizzo di memoria dell’istruzione da eseguire. Una volta prelevata l’istruzione il PC viene incrementato per puntare all’operazione successiva. L’indirizzo dell’istruzione viene copiato del MAR e successivamente nel BUS ADDR. Dopo un’opportuna decodifica dell’indirizzo si trova la cella di memoria, e si preleva l’istruzione. L’istruzione prelevata viene inserita nel BUS DATI e finirà nel MDR.
* **Decode istruzione:** L’istruzione entra nell’IR(Instruction Register) e viene interpretata dalla CU. Vengono preparate le risorse necessarie. L’istruzione deve interfacciarsi direttamente con l’Hardware in base a delle regole definite dalla logica in uso:
  + **Logica cablata:** l’HW è predisposto a compiere l’istruzione seguendo i passaggi determinati dalla struttura fisica dei componenti. Più complicato e costoso da realizzare, e meno flessibile. Offre prestazioni migliori
  + **Logica microprogrammata:** l’insieme di microistruzioni da compiere per effettuare l’operazione fisicamente è storata all’interno di una ROM adatta. La CU è a sua volta una microarchitettura capace di eseguire delle istruzioni programmate. Meno costoso e più versatile, ma meno efficiente
* **Fetch degli operandi:** dopo la decodifica dell’istruzione si cercano gli eventuali operandi, tramite gli indirizzi di memoria espressi nell’istruzione decodificata. Ancora una volta entra in gioco in BUS ADDR e il MAR per l’indirizzo di memoria del dato da cercare, e una volta trovato il BUS DATI lo mette nel MDR.Affinchè l’operazione possa essere eseguita gli operandi devono trovarsi all’interno dei registri della CPU
  + 4 metodi di indirizzamento:
    - **Immediato:** il dato su cui operare è gia presente nell’istruzione, quindi può essere una costante numerica o letterale
    - **Diretto:** viene fornito l’indirizzo di memoria in cui andare a cercare il dato su cui operare
    - **Indiretto:** fornire l’indirizzo della cella di memoria contenente a sua volta un indirizzo di memoria che punta alla cella effettiva in cui si trova il dato
    - **Base + Offset:** all’interno dell’istruzione è presente un registro GPR(General Purpose Register) che conterrà l’indirizzo base. A questa base viene sommato un offset in modo da trovare la cella corretta.
* **Execute operazione:** se l’operazione è un calcolo viene eseguita dalla ALU, altrimenti se è una lettura in memoria oppure un salto viene gestita diversamente. Il risultato viene salvato in un registro, che poi sarà opportunamente trasferito in memoria
* **Write Back:** vengono ripristinati i registri usati dall’operazione, e in particolare viene aggiornato il PSW

DATA PATH



REGISTRI -> MEMORIA

 Immagine che contiene tavolo

Descrizione generata automaticamente

Registri: 32 bit di larghezza, 32 registri totali.

Memoria: 8 bit larghezza, 2^nbit nbit=bit BUS ADDR

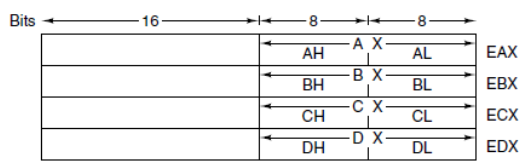
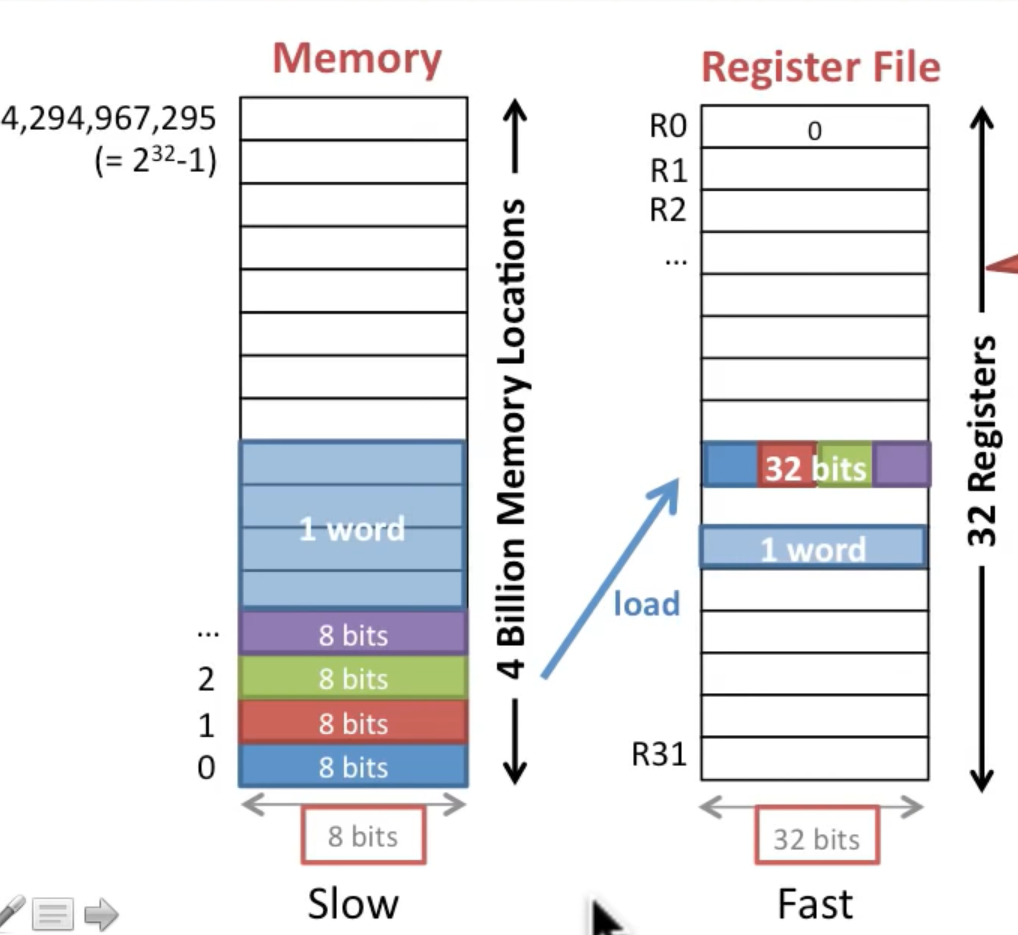


Immagine che contiene testo

Descrizione generata automaticamente



4 celle di memoria vengono salvate in 1 registro

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene tavolo

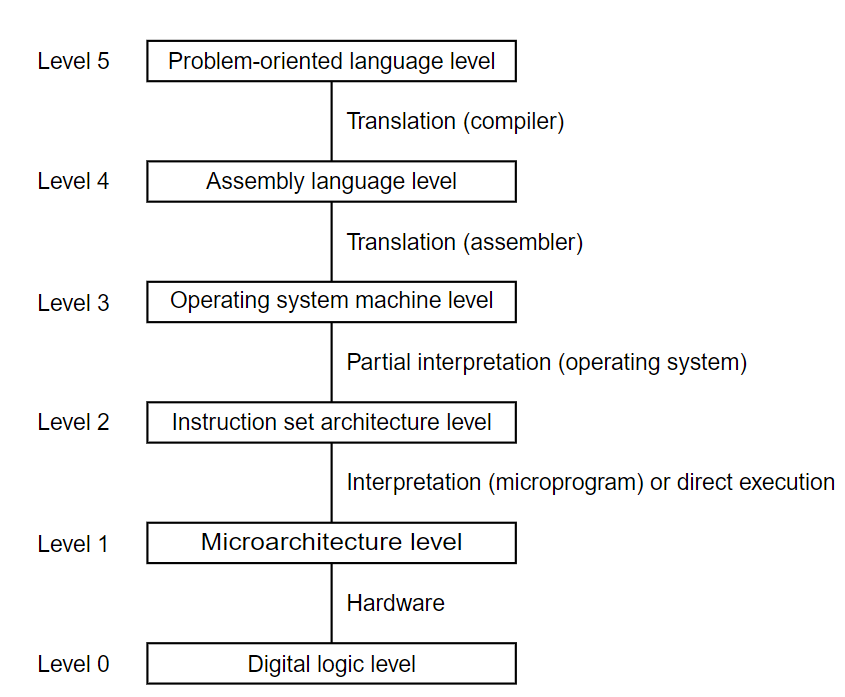
Descrizione generata automaticamente

Ogni indirizzo identifica 4 celle di memoria, che verranno tutte trasferite ai registri.

Essendo multipli di 4, gli indirizzi termineranno sempre con 00

ACCESSO ALLA RAM

* Allineato: viene prelevata l’intera riga di celle, quindi l’indirizzo sarà sempre multiplo di 4
* Non allineato: viene indicato un indirizzo intermedio. Vengono sempre prelevate 4 celle, anche appartenenti alla riga successiva. Essendo più complicato da realizzare solitamente si prelevano le 2 righe intere e poi vengono frammentate nelle celle interessate



**ISA (Instruction Set Architecture) :** insieme delle istruzioni di basso livello che operano nel livello HW .

Interfaccia il LVL1 (HW):

* Deve coprire tutte le funzioni disponibili per il CPU
* Tutte le istruzioni eseguibili
* Tipi di dato comprensibili alla CPU: VEDI CODIFICA DATI

Immagine che contiene testo

Descrizione generata automaticamente

* Modello di memoria RAM
* FORMATO ISTRUZIONE
* Modalità di Indirizzamento

con LVL3 (SW):

* Modalità di funzionamento
* Gestione Input Output

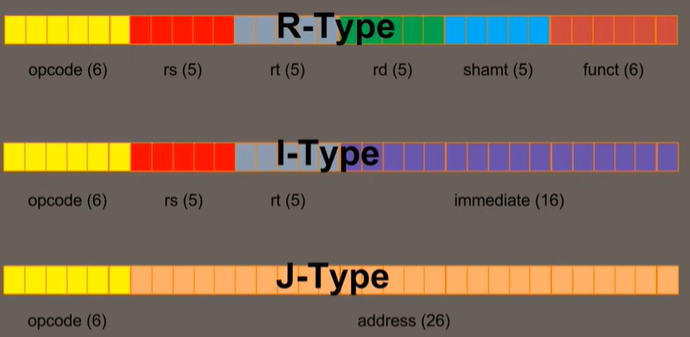
COMPORTAMENTO:

* Formale: in base alla istruzione il comportamento dell’HW è ben definito e costante
* Informale: Garantisce la retrocompatibilità, quindi in base alla struttura HW esegue le istruzioni

A seconda della architettura della CPU, RISC o CISC sono individuati dei set più o meno grandi e complessi di istruzioni. VLIW (Very Long Instruction Word).

Le istruzioni devono:

* Sfruttare HW a max velocità e min costo
* Robuste e a prova di errore
* Operazioni ALU
* Poca memoria accessibile durante l’esecuzione (registri CPU)
* Valutare semplici condizioni
  + - **Operative: le operazioni vengono eseguite dalla ALU**
      * **Operazione aritmetica**
      * **Operazione logica**
    - **Trasferimento dei dati da MEM a CPU e viceversa**
      * **Load word:** carica i dati dalla RAM ai registri
      * **Store word:** carica i dati dai registri alla RAM
    - **Controllo:** variazione dell’ordine di esecuzione di istruzioni. SALTI
      * **Rotazione o SHIFT**
      * **Aggiornamento flag di stato (PSW)**
      * **Controllo del flusso del programma: chiamate a procedure utente**
      * **Controllo della macchina: procedure di sistema e interrupt**
      * **INPUT/OUTPUT**
  + **Formato istruzione 32bit** [MIPS ISA refs](https://inst.eecs.berkeley.edu/~cs61c/resources/MIPS_help.html)



* **R TYPE:** operazioni aritmetico logiche, da assegnare alle ALU
  + **Opcode (0 or 1):** 6 bit che indicano di leggere il campo **funct**
  + **RS: Source Register:** 1 operando
  + **RT :** 2 operando
  + **RD :** **Destination Register**: dove va salvato il risultato
  + **Shift amount**: usato nelle operazioni di shifting per indicare di quanto traslare una sequenza di bit
  + **Funct**: campo che contiene il codice univoco del tipo di operazione da fare eseguire alle ALU.
* **I TYPE:** operazioni di accesso in memoria (salti condizionati)oppure contenenti delle costanti (indirizzamento immediato)
  + **Opcode:** 6 bit che indicano il tipo di istruzione
  + **RS:** Source Register:
  + **RT:** registro di destinazione
  + **IMM:** valore numerico in 16 bit. Deve essere esteso a 32 bit durante la fase di Instruction Decode **(Sign-Extended)**
* **J TYPE:** istruzioni di salto JUMP incondizionato

Immagine che contiene tavolo

Descrizione generata automaticamente

**OPERATION CODE , DEST, SRC1, SRC2: SRC indica il dato su cui operare e DEST indica il registro in cui salvare il risultato**

**CLASSE**: 3 bit che identificano la classe dell’op code

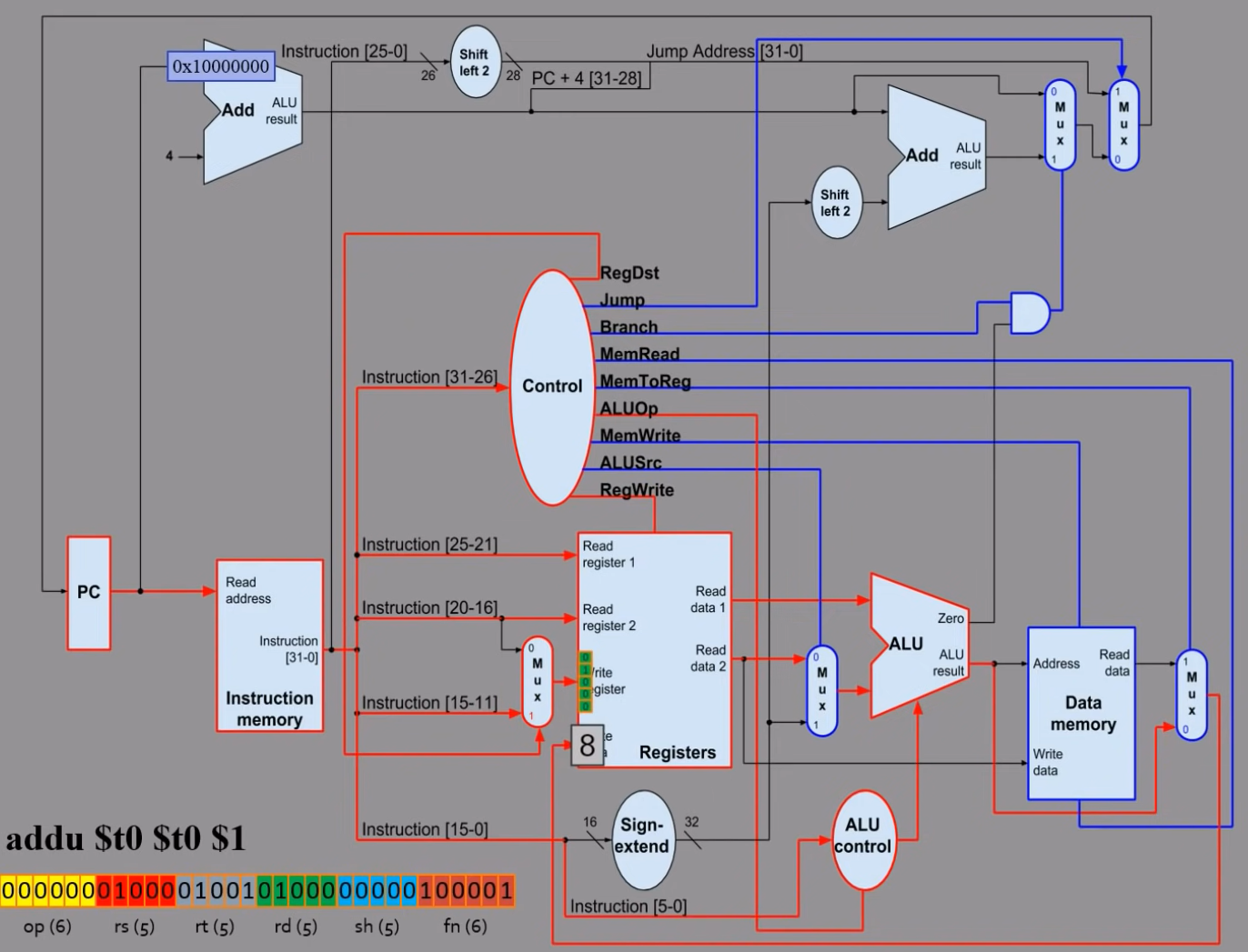
* 000 = movimento dati
* 001 = Operazione aritmetica
* 010 = Operazione logica
* 011 = Rotazione e Shift
* 100 = Operazione sui flag di stato PSW
* 101 = Controllo del flusso del programma. SALTI, LOOP, Branches
* 110 = Controllo della macchina
* 111 = I/O

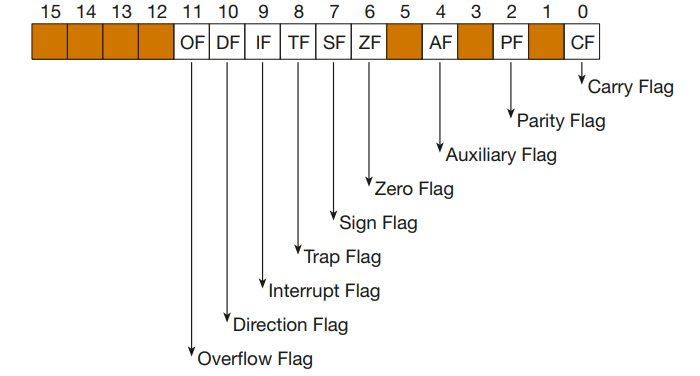
Immagine che contiene tavolo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente



* **PC** contiene l’indirizzo di memoria della prossima istruzione da eseguire.
* **Instruction register:** contiene l’istruzione appena prelevata dalla memoria dall’indirizzo indicato dal PC. Una volta che l’istruzione è stata caricata il PC viene incrementato
* **OPCODE:** viene inserito nella Control Unit, che attiva dei segnali di controllo, che viaggeranno nel BUS controllo, i quali coordineranno il funzionamento dal datapath e preparano il circuito a eseguire correttamente l’istruzione
  + Nel caso in esame l’operazione da effettuare è un ADD unsigned, quindi i flags attivati dalla CU sono ALUOP, REGWRITE (se si deve scrivere il risultato in un registro) e RESDEST (indica che la destinazione del risultato è un registro)
* **RS e RT:** inseriti nei Read Register-1 e 2. Questi 2 registri saranno i 2 operandi della ALU
* **RGDST:** da un segnale di enable al MUX, indicando di inserire il RD nel campo Write Register, quindi il registro di destinazione del risultato.
  + **Registro dei Flags. FLAG = 1bit di controllo.** Determinano degli stati della CPU. = CCR (Condition Code Register)   
    = PSW (Program Status Word)
    - * **Overflow:** segnala il verificarsi di un overflow al seguito di una operazione aritmetica
      * **Zero:** segnala se il risultato dell’operazione è 0
  + **REGISTRI DEI DATI**
    - **MDR (Memory Data Register):** Collegato al **BUS DATI** tramite **buffer bidirezionale a 3 stati.** Contiene i dati ricevuti o da inviare alla memoria.

Da MEM a CPU i dati finiscono nei registri interni

Da CPU a MEM i dati partono dai registri e arrivano in RAM

* + - **MAR (Memoria Address Register):** Nella fase di fetch contiene l’indirizzo di memoria dell’istruzione, che poi verrà inserito nel IR. Nella fase di execute contiene l’indirizzo di memoria del dato da cercare (operando dell’istruzione).   
      Collegato al **BUS ADDR**