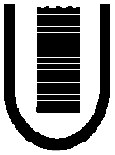
Architettura dei Calcolatori
Prof. Enrico Nardelli

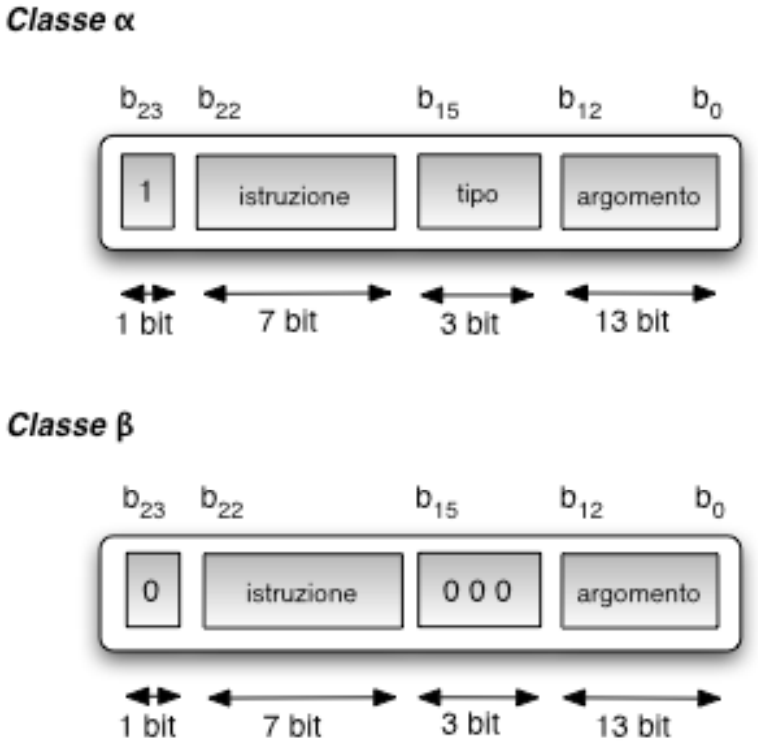


Università degli Studi di Roma “Tor Vergata”

Virtual CPU (Eniac): parte 3

Classi di istruzioni

- La differenza tra le due categorie viene formalizzata per mezzo della specifica di due classi di istruzioni: la classe α e la classe β.
- Le due classi si distinguono per mezzo dell'MSB dell'istruzione: il bit b_{23}.
- Nella classe β i bit b_{15}...b_{13} sono sempre assegnati a zero in quanto inutilizzati. La tipologia dell'argomento è insita nel codice operativo.



Il formato delle istruzioni di vCPU

Presentiamo in dettaglio il formato delle istruzioni e i criteri usati per assegnare gli opcode alle varie operazioni

Definiamo formalmente i termini *argomento* e *operando*

- Argomento: e' il valore dei bit $b_{12}...b_0$ presenti nell'istruzione;
- Operando: e' il valore effettivo utilizzato dall'operazione intrapresa dall'istruzione

Come visto le istruzioni hanno un argomento che può essere di tipo diverso

Il formato delle istruzioni di vCPU

In vCPU ci sono 5 tipi di argomento :

- **immediato**: l'operando coincide con l'argomento.
- **registro**: l'operando è il valore contenuto nel registro specificato come argomento.
- **indirizzo diretto**: l'operando è il valore contenuto nella cella di memoria specificata come argomento.

Il formato delle istruzioni di vCPU

- **indirizzo indiretto**: l'operando è il valore contenuto nella cella di memoria il cui indirizzo e' contenuto nella cella di memoria specificata come argomento.
- **registro indiretto**: l'operando è il valore contenuto nella cella di memoria il cui indirizzo è contenuto nel registro specificato come argomento.

Il prefisso "@" indica che il valore effettivo usato dall'istruzione non corrisponde direttamente all'argomento, ma si ottiene da esso.

Il formato delle istruzioni di vCPU

Esempi di questi cinque tipi di argomenti, per l'istruzione ADD che somma il valore dell'accumulatore a quello specificato dall'argomento, sono:

- **Immediato**: ADD 100
- **Indirizzo diretto**: ADD @100
- **Indirizzo indiretto**: ADD @@100
- **Registro**: ADD BX
- **Registro indiretto**: ADD @BX

Ricordiamo che il registro accumulatore in vCPU è il registro AX

Le istruzioni di vCPU

- Sono suddivise nelle seguenti categorie, prescindendo dalla loro classe, per essere analizzate in base alla loro funzione:
  - **Memorizzazione** sono le istruzioni che permettono il flusso di dati dai registri alla memoria e viceversa.
  - **Aritmetico-logiche** sono le istruzioni che effettuano operazioni sui dati: sono sia aritmetiche che logiche.
  - **Controllo del flusso del programma** sono le istruzioni che possono deviare il flusso del programma.
  - **I/O** sono le istruzioni che permettono il flusso dei dati da e verso l'esterno

Istruzioni di memorizzazione

- **LOAD** *argomento*
  - Copia nell'accumulatore AX il valore specificato dall'argomento, che viene indicato con la sintassi presentata prima
  - Il precedente valore contenuto in AX viene perso
  - I flag aritmetici e logici **non vengono influenzati**

- **STORE** *argomento*
  - Copia il valore contenuto nell'accumulatore AX nella destinazione specificata dall'argomento, che viene indicato con la sintassi presentata prima
  - **Non è ammesso** un argomento immediato
  - Il contenuto di AX rimane inalterato
  - I flag aritmetici e logici **non vengono influenzati**

Istruzioni di memorizzazione
Primi esempi

Scrivere le seguenti istruzioni:

- Caricare nel registro AX il numero 25
- Caricare nel registro AX il contenuto della cella 7
- Caricare nel registro AX il contenuto della cella specificata nella cella 7
- Caricare nel registro AX il contenuto del registro CX
- Caricare nel registro AX il contenuto della cella specificata nel registro CX

- Caricare nel registro CX il contenuto della cella specificata nella cella 7

Istruzioni di memorizzazione
Primi esempi - soluzioni

- Caricare nel registro AX il numero 25

  -> LOAD 25
- Caricare nel registro AX il contenuto della cella 7

  -> LOAD @7
- Caricare nel registro AX il contenuto della cella specificata nella cella 7

  -> LOAD @@7
- Caricare nel registro AX il contenuto del registro CX

  -> LOAD CX
- Caricare nel registro AX il contenuto della cella specificata nel registro CX

  -> LOAD @CX
- Caricare nel registro CX il contenuto della cella specificata nella cella 7

  -> LOAD @@7

  STORE CX

Istruzioni aritmetico-logiche – 1

- Ci sono 8 istruzioni aritmetiche, di cui 6 manipolano l'accumulatore AX
  - **ADD** *argomento*, per l'addizione
  - **SUB** *argomento*, per la sottrazione
  - **MUL** *argomento*, per la moltiplicazione
  - **DIV** *argomento*, per la divisione
  - **MOD** *argomento*, per l'operazione di modulo (calcolo del resto)
  - **NEG**, per effettuare il complemento a 2 di AX (*non ha argomenti*)
  - Effettua l'operazione aritmetica tra il valore contenuto nell'accumulatore AX ed il valore specificato dall'argomento, che viene indicato con la sintassi presentata prima
  - Il risultato viene memorizzato in AX e il precedente valore contenuto in AX viene perso
  - I flag aritmetici e logici vengono influenzati

Istruzioni aritmetico-logiche – 2

- Altre 2 istruzioni aritmetiche manipolano direttamente l'argomento
  - **INC** *argomento*, per l'addizione unitaria (incremento di 1)
  - **DEC** *argomento*, per la sottrazione unitaria (decremento di 1)
  - Effettua l'operazione aritmetica sul valore specificato dall'argomento, che viene indicato con la sintassi presentata prima
  - **Non è ammesso** un argomento immediato
  - Il risultato viene memorizzato nella destinazione specificata dall'argomento e il precedente valore contenuto in esso viene perso
  - I flag aritmetici e logici vengono influenzati

Istruzioni aritmetico-logiche – 3

- Ci sono 4 istruzioni logiche
  - **AND** *argomento*, per l'operazione di AND logico
  - **OR** *argomento*, per l'operazione di OR logico
  - **XOR** *argomento*, per l'operazione di XOR logico (OR esclusivo)
  - **NOT**, per l'operazione di negazione logica bit-a-bit (*senza argomento*)
  - Effettua l'operazione aritmetica tra il valore contenuto nell'accumulatore AX ed il valore specificato dall'argomento, che viene indicato con la sintassi presentata prima
  - Il risultato viene memorizzato in AX e il precedente valore contenuto in AX viene perso
  - I flag aritmetici e logici vengono influenzati

Istruzioni aritmetico logiche – 4

- Come interpretare correttamente dal punto di vista aritmetico il valore dei flag complessi.
- Ricordiamo preliminarmente come la ALU assegna i valori 0 e 1 a questi flag
- Nel caso di uso dell'Adder
  - CA=1 se il bit di riporto dell'Adder vale 1 nell'ultima operazione, CA=0 altrimenti.
  - OV=1 se l'ultima operazione ha generato un risultato il cui bit più significativo differisce da entrambi i bit più significativi degli operandi (tra loro uguali), OV=0 altrimenti;
- Nel caso di uso del Multiplier
  - OV=0 quando i bit in R1 e il bit più significativo in R0 sono tutti uguali tra di loro, OV=1 altrimenti.
  - CA: assegnato con lo stesso criterio usato per OV.

Istruzioni aritmetico logiche – 5

- ADD, SUB, INC, DEC
  - **OV** : Se viene assegnato ad 1 indica che la somma ha generato un risultato troppo grande, in modulo, per essere memorizzato in una parola (overflow) e pertanto non è possibile fruirne. Inoltre, quando questo flag è assegnato ad 1, il valore degli altri flag semplici perde di significato.
  - **CA** : Nel caso di operazioni su valori rappresentati in notazione ordinaria (cioè non complementata) indica anch'esso una condizione di overflow. Inoltre, il numero ottenuto affiancando questo bit, come il più significativo, al risultato della ALU in AX, rappresenta sempre il risultato anche in condizioni di overflow.

Istruzioni aritmetico logiche – 6

- MUL
  - **OV** : viene assegnato ad 1 se il risultato è contenuto in BX:AX, a 0 se è stato possibile memorizzarlo solo su AX. Non esistendo overflow per questa operazione, il significato che viene associato a questo flag è quello di overflow relativamente alla dimensione di una parola.
  - **CA** : il significato aritmetico di questo flag è lo stesso di OV.

Istruzioni di controllo del flusso – 1

- Ci sono 5 istruzioni di manipolazione dei flag
  - **CMC**, per complementare il flag CA (carry)
  - **CMO**, per complementare il flag OV (overflow)
  - **CMS**, per complementare il flag SI (segno)
  - **CMP**, per complementare il flag EV (parità)
  - **CMZ**, per complementare il flag ZE (zero)

  - Effettua la negazione logica del flag specificato dall'istruzione
  - Influenza solo il flag che viene complementato

Istruzioni di controllo del flusso – 2

- C’è un’istruzione di salto incondizionato
  - **JMP** *argomento*, per saltare all’indirizzo specificato dall’argomento
  - È ammesso solo un argomento immediato
- Ci sono 10 istruzioni di salto condizionato al valore dei flag
  - **J*X*** *argomento*, per saltare all’indirizzo specificato dall’argomento se il valore del flag ***X*** è 1
  - **JN*X*** *argomento*, per saltare all’indirizzo specificato dall’argomento se il valore del flag ***X*** è 0
  - ***X*** può essere ***C*** (flag CA), ***O*** (OV), ***S*** (SI), ***P*** (EV), ***Z*** (ZE)
  - È ammesso solo un argomento immediato
  - I flag aritmetici e logici **non vengono influenzati**

Istruzioni di controllo del flusso – 3

- Ci sono 4 istruzioni di salto condizionato al valore di test complessi che coinvolgono il valore di più flag
  - **J*X*** *argomento*, per saltare all'indirizzo specificato dall'argomento se il valore del test specificato da ***X*** è 1
  - **J*X*E** *argomento*, per saltare all'indirizzo specificato dall'argomento se il valore del del test specificato da ***X*** è 0
  - ***X*** può essere ***A*** (), ***B*** (), ***G*** (), ***L*** ()
  - È ammesso solo un argomento immediato
  - I flag aritmetici e logici **non vengono influenzati**
  - VA RICOSTRUITO IL SIGNIFICATO DEI TEST. Nella tesi di Codella c'è questo passaggio
  - Nel caso dei test complessi (b20 = 1), invece, la situazione `e la seguente:
  - b20 b19 b18 b17 b16

Istruzioni di controllo del flusso – 4

- VA RICOSTRUITO IL SIGNIFICATO DEI TEST. Nella tesi di Codella c'è questo passaggio
- Nel caso dei test complessi (b20 = 1), invece, la situazione `e la seguente:
- b20 b19 b18 b17 b16
- Salta se CA=0 e ZE=0 1 0 0 0 0
- Salta se CA=1 e ZE = 1 1 0 0 1 1
- Salta se ZE = 0 e SI = OV 1 0 1 0 0
- Salta se SI = OV 1 0 1 0 1
- Salta se SI != OV 1 1 1 1 0
- Salta se ZE = 1 e SI != OV 1 1 1 1 1
- Bisogna notare che in realt`a esistono dei test complessi che coincidono con quelli semplici: in particolar modo, pi`u avanti vedremo due istruzioni di salto complesse, JAE e JB, le cui condizioni coincidono rispettivamente con quelle di JNC e JC. Per mantenere entrambe le coppie di salto senza per`o compromettere l'unicit`a della codifica dell'istruzione, vengono assegnate le stesse codifiche ai salti JAE, JNC e a JB, JC.
- Vedere anche la tabella con il repertorio delle istruzioni di VCPU

Istruzioni di ingresso-uscita

- Ci sono 2 istruzioni di ingresso-uscita
  - **IN** *argomento*, per l'ingresso
    - L'argomento è un valore che indica una delle 2^{12} porte di ingresso-uscita
    - Il valore letto nella porta specificata dall'argomento viene copiato in AX e il precedente valore contenuto in AX viene perso
  - **OUT** *argomento*, per l'uscita
    - L'argomento è un valore che indica una delle 2^{12} porte di ingresso-uscita
    - Il valore contenuto nell'accumulatore AX viene copiato nella porta specificata dall'argomento. Il valore contenuto in AX viene mantenuto.
  - Per entrambe le istruzioni flag aritmetici e logici **non vengono attivati**

Le istruzioni di vCPU – 1

- Presenteremo una tabella per ogni categoria di istruzione.
- Le tabelle sono strutturate nel seguente modo :
  1. codice macchina dell'istruzione
  2. sintassi
  3. semantica scritta in un semplice formalismo
  4. flag che possono essere modificati
  5. un esempio

Le istruzioni di vCPU – 2

- In tali tabelle sono contenute tutte le possibili varianti di una istruzione relativamente al tipo di argomento.
- Nel presentare e discutere le istruzioni useremo le seguenti convenzioni :
  - **R** : per indicare un nome di un registro
  - **N** : per indicare un intero positivo in complemento a 2 sui 13 bit dell'argomento, appartenente quindi all'intervallo [0, +4095]
  - **Z** : per indicare un intero relativo in complemento a 2 sui 13 bit dell'argomento, appartenente quindi all'intervallo [−4096, +4095].

Le istruzioni di vCPU – 3

- QUI BISOGNA PRESENTARE/INSERIRE LE TABELLE

Il formalismo per la semantica delle istruzioni – 1

- Per specificare il significato (o semantica) dell'istruzione è utilizzato un semplice formalismo che specifica un'azione costituita da due fasi.
  - computare una espressione matematica che usa gli operatori tradizionali
  - assegnare ovvero memorizzare il risultato dell'espressione in un luogo in grado di contenere il dato (registri o memoria)

Il formalismo per la semantica delle istruzioni – 2

- E’ utilizzata la seguente sintassi

$$expr \rightarrow dest$$

expr : è l’espressione da computare

dest : è la destinazione del risultato della computazione

Il formalismo per la semantica delle istruzioni – 3

- In alcuni casi l'azione può essere preceduta da una condizione, con l'interpretazione che l'azione viene eseguita solo quando la condizione è verificata

Si parla allora di *azione condizionata*

$$cond \Rightarrow espr \rightarrow dest$$

- *Se* non si verifica la condizione *cond* l'azione associata all'istruzione non viene eseguita

Il formalismo per la semantica delle istruzioni – 4

- In questo formalismo adottato per mostrare la semantica le parentesi tonde sono usate per descrivere le modalità di indirizzamento ad una cella di memoria

Esempi :

- 123 indica il valore 123
- (123) indica il contenuto della cella di indirizzo 123
- ((123)) indica il contenuto della cella il cui indirizzo è contenuto nella cella 123

Il formalismo per la semantica delle istruzioni – 5

- In questo formalismo quando indichiamo il nome di un registro intendiamo riferirci al suo valore
- Ma il nome di un registro tra parentesi viene interpretato secondo la convenzione sopra specificate

Esempi :

- se il registro BX contiene il valore 123, allora 27+BX indica la somma tra 27 e 123
- 27 + (BX) indica la somma fra 27 ed il contenuto della cella di memoria 123
- (27) + (BX) indica la somma tra il contenuto della cella 27 e quello della cella 123

Il formalismo per la semantica delle istruzioni – 6

- Fare attenzione in questo formalismo quando celle e registri sono i destinatari delle assegnazioni
  - L'assegnazione di un valore ad una cella di memoria si specifica indicando il suo indirizzo tra parentesi
  - L'assegnazione di un valore ad un registro si specifica indicandone il nome

Esempi :

- STORE @27 ha come semantica AX → (27), cioè il valore contenuto nell'accumulatore viene scritto nella cella 27
- STORE BX ha come semantica AX → BX, cioè il valore contenuto nell'accumulatore viene scritto nel registro BX
- STORE @BX ha come semantica AX → (BX), cioè il valore contenuto nell'accumulatore viene scritto nel registro BX

Il formalismo per la semantica delle istruzioni – 7

- Per indicare che un indirizzo si riferisce alle porte di I/O piuttosto che alla memoria, vengono usate delle parentesi quadre attorno al numero di porta
  - Con [N] ci si sta riferendo alla porta N
- Quando si ha necessità di riferirsi contemporaneamente a due registri si utilizza l'operatore ":"
  - il registro che segue l'operatore memorizza la porzione più bassa, cioè meno significativa, del dato
  - il registro che precede l'operatore memorizza quella più alta, cioè più significativa (non è possibile inserirla nel primo)
  - Esempio: BX : AX

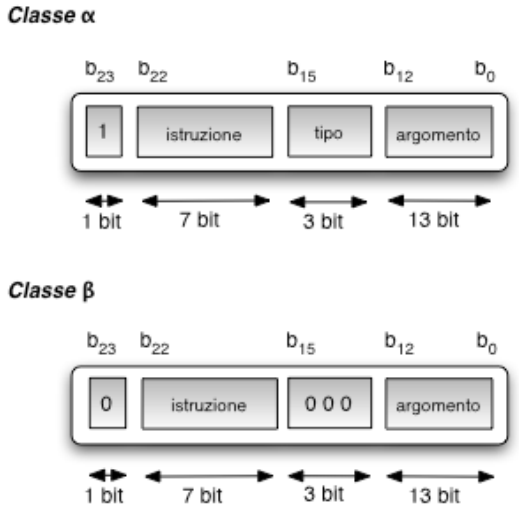
Il formalismo per la semantica delle istruzioni – 8

- Le parentesi graffe vengono utilizzate per raccogliere gli operandi dei predicati logici di primo ordine
  - **AND{x,y}** Indica l'AND logico applicato bit a bit agli argomenti x e y
  - **OR{x,y}** Indica l'OR logico applicato bit a bit agli argomenti x e y
  - **XOR{x,y}** Indica l'OR esclusivo (XOR) logico applicato bit a bit agli argomenti x e y
  - **NOT{x}** Indica il NOT logico applicato ad ogni singolo bit dell'argomento x

- Per la semantica che esce dal campo d'azione del formalismo adottato, verrà utilizzata la lingua italiana

Criterio di assegnazione degli opcode alle istruzioni

- I codici di operazione sono stati assegnati alle istruzioni in modo razionale e che sia facile ricordare quale sia il codice operativo associato una data istruzione.
- Descriviamo il criterio di assegnazione degli opcode in relazione sia alla classe α che alla classe β.
- Le due classi si distinguono per mezzo dell'MSB dell'istruzione: il bit b_{23}



Istruzioni nella classe α (argomento di tipo variabile) Codifica delle operazioni

- b_{23}=1 e i bit di tipo b_{15}...b_{13} distinguono il tipo di argomento
- 2^7 istruzioni differenti divise nelle seguenti categorie:
  - Prodotti
  - Somme
  - Memorizzazione
  - Logiche ed altro.
- Dei 7 bit disponibili per codificare l'istruzione dedichiamo
  - $b_{22}b_{21}$ a rappresentare la categoria
  - b_{20}...b_{16} a codificare la specifica operazione all'interno della categoria

Istruzioni nella classe α (argomento di tipo variabile) Codifica delle operazioni

- In realtà, dato il numero relativamente ridotto di istruzioni appartenenti a questa categoria, i bit dedicati realmente alla codifica di queste sono solamente i bit $b_{20}b_{19}$ e i bit $b_{18}...b_{16}$ sono sempre configurati a 0

- I bit $b_{22}b_{21}$ codificano la categoria come illustrato in tabella

	b_{22}	b_{21}
Prodotti	1	1
Somme	1	0
Memorizzazione	0	1
Logiche ed altro	0	0

- Notare che $b_{22} = 1$ contraddistingue istruzioni aritmetiche

Istruzioni nella classe α (argomento di tipo variabile) Codifica delle operazioni

- La codifica delle specifiche operazioni per ognuna delle categorie, realizzata mediante i bit $b_{20}b_{19}$, avviene inoltre come illustrato dalle seguenti tabelle.

Prodotti

	b_{20}	b_{19}
DIV	1	1
MUL	1	0

Somme

	b_{20}	b_{19}
INC	1	1
DEC	1	0
ADD	0	1
SUB	0	0

Solo b_{19} contraddistingue le due istruzioni

b20 = 1 solo per oper. elementari INC e DEC

Memorizzazione

	b_{20}	b_{19}
ST	1	1
LD	0	0

Logiche ed altro

	b_{20}	b_{19}
AND	1	1
OR	1	0
XOR	0	1
MOD	0	0

Solo il modulo ha entrambi i bit a 0

(non è un'istruzione logica)

Istruzioni nella classe α (argomento di tipo variabile) Codifica dei registri

- In molte istruzioni si ha la possibilità di specificare un registro come argomento. Per fare ciò è necessario assegnare una codifica ad ognuno di esso.
- Codifica adottata da vCPU

	b_{12}	$b_{11}...b_3$	b_2	b_1	b_0
AX	0	0...0	0	0	0
BX	0	0...0	0	0	1
CX	0	0...0	1	1	0
DX	0	0...0	1	1	1
PC	0	1...1	1	1	1

- Il registro PSW non ha una codifica in quanto non può essere usato come operando di una istruzione

Istruzioni nella classe α (argomento di tipo variabile) Codifica del tipo di argomento

- Come detto la tipologia dell'argomento delle istruzioni di tipo α viene specificata dai 3 bit di tipo b_{15}...b_{13}
- b_{15} identifica la natura dell'argomento dell'istruzione :
  - 1 indica che l'argomento è un nome di registro
  - 0 indica che è un numero naturale

- Gli altri due bit, $b_{14}b_{13}$, assumono di conseguenza significati diversi in base alla natura dell'argomento

Istruzioni nella classe α (argomento di tipo variabile) Codifica del tipo di argomento

- $b_{15} = 1$ (l'argomento è un nome di registro)
  - $b_{14} = 0$, $b_{13} = 0$: il registro contiene l'operando (argomento di tipo registro).
  - $b_{14} = 0$, $b_{13} = 1$: il registro contiene l'indirizzo della cella in cui è contenuto l'operando (argomento di tipo registro indiretto)
- $b_{15} = 0$ (l'argomento è un numero naturale)
  - $b_{14} = 0$, $b_{13} = 0$: il numero è l'operando (argomento di tipo immediato): è quindi un numero relativo.
  - $b_{14} = 0$, $b_{13} = 1$: il numero è l'indirizzo della cella che contiene l'operando (argomento di tipo indirizzo diretto): in questo caso si tratta invece di un numero positivo
  - $b_{14} = 1$, $b_{13} = 0$: il numero è l'indirizzo della cella che contiene l'indirizzo della cella contenente l'operando (argomento di tipo indirizzo indiretto): anche in questo caso è un numero positivo

Istruzioni nella classe β (argomento di tipo prefissato)

- b_{23}=0 e i bit di tipo, b_{15}...b_{13} tutti sempre a 0

- 2^7 istruzioni differenti divise in quattro categorie:
  - I/O
  - Aritmetico-logiche
  - Salto
  - Altro

Istruzioni nella classe β (argomento di tipo prefissato)

- Dei 7 bit disponibili per codificare l'istruzione i bit $b_{22}b_{21}$ codificano la categoria nel seguente modo :

	b_{22}	b_{21}
I/O	1	1
Aritmetico-logiche	1	0
Salto	0	1
Altro	0	0

- i bit b_{20}...b_{16} codificano le operazioni specifiche di ogni categoria nelle prossime tabelle vedremo in che modo

Istruzioni nella classe β (argomento di tipo prefissato)

- **I/O**

	b_{20}	b_{19}	b_{18}	b_{17}	b_{16}
IN	0	0	0	0	1
OUT	0	0	0	0	0

Le due sole istruzioni di questa categoria sono distinte dal valore di b_{16}. Per entrambe queste istruzioni, l'argomento è sempre un numero naturale che identifica il numero di porta sulla quale operare

- **Aritmetico-logiche**

	b_{20}	b_{19}	b_{18}	b_{17}	b_{16}
NOT	0	0	0	0	1
NEG	0	0	0	0	0

Le due sole istruzioni di questa categoria sono distinte dal valore di b_{16}. Come si vedrà meglio in seguito NOT è l'operazione booleana che complementa l'argomento bit per bit, mentre NEG è l'operazione aritmetica che nega l'argomento

Istruzioni nella classe β (argomento di tipo prefissato)

- **Salto**

Questa categoria conta un numero di istruzioni più alto : per questo motivo è bene suddividere i bit $b_{20}...b_{16}$ in tre parti :

- prima parte è composta dal solo bit b_{20}, per identificare il tipo di salto;
- seconda parte è composta dai bit $b_{19}...b_{17}$ per identificare il tipo di test
- la terza, composta dal bit b_{16}, per identificare la modalità del test

Istruzioni nella classe β (argomento di tipo prefissato)

- Possiamo distinguere due tipi di salto :
  - SALTO SEMPLICE : è quello condizionato dal valore di un singolo flag
  - SALTO COMPLESSO : è quello condizionato dal valore contemporaneo di più flag.

- Si possono distinguere i due tipi di salto guardando il bit b_{20} :

- se è a 0 si sta codificando un salto semplice
- se è a 1 si sta codificando un salto complesso

Istruzioni nella classe β (argomento di tipo prefissato)

- Nel caso di salto semplice ($b_{20} = 0$) abbiamo le seguenti istruzioni, in base al valore dei bit $b_{19}...b_{16}$

	b_{20}	b_{19}	b_{18}	b_{17}	b_{16}
Salta sempre	0	0	0	0	0
Salta se CA = 1	0	0	0	1	1
Salta se CA = 0	0	0	0	1	0
Salta se OV = 1	0	0	1	0	1
Salta se OV = 0	0	0	1	0	0
Salta se SI = 1	0	0	1	1	1
Salta se SI = 0	0	0	1	1	0
Salta se EV = 1	0	1	0	0	1
Salta se EV = 0	0	1	0	0	0
Salta se ZE = 1	0	1	0	1	1
Salta se ZE = 0	0	1	0	1	0

- L' effetto del bit b_{16} nei salti semplici è correlato alla condizione di salto
  - Se b_{16} è = 1, allora il salto avverrà qualora il flag relativo avrà valore 1: analogamente,
  - se b_{16} è = 0, allora il salto avverrà solo se il flag relativo è configurato a 0

Istruzioni nella classe β (argomento di tipo prefissato)

- Veniamo ora al caso dei test complessi ($b_{20} = 1$)

	b_{20}	b_{19}	b_{18}	b_{17}	b_{16}
Salta se CA=0 e ZE=0	1	0	0	0	0
Salta se CA=1 e ZE = 1	1	0	0	1	1
Salta se ZE = 0 e SI = OV	1	0	1	0	0
Salta se SI = OV	1	0	1	0	1
Salta se SI != OV	1	1	1	1	0
Salta se ZE = 1 e SI != OV	1	1	1	1	1

- Si noti che esistono dei test complessi che coincidono con quelli semplici :
  - JAE e JB sono due istruzioni di salto complesse le cui condizioni coincidono rispettivamente con quelle di JNC e JC.
  - Per mantenere entrambe le coppie di salto senza compromettere l'unicità della codifica dell'istruzione, vengono assegnate le stesse codifiche ai salti JAE, JNC e a JB, JC.

Istruzioni nella classe β (argomento di tipo prefissato)

- Ultima categoria della classe comprende due sottocategorie
  - La prima contiene due istruzioni che non possono essere classificate in nessuna categoria analizzata precedentemente,
  - la seconda contiene alcune istruzioni inserite in questa categoria per motivi di opportunità.
- Nessuna di queste istruzioni ha un argomento

Istruzioni nella classe β (argomento di tipo prefissato)

- In particolare, la prima sottocategoria (contraddistinta da $b_{20} = b_{19} = 0$) contiene le istruzioni elencate di seguito

	b_{20}	b_{19}	b_{18}	b_{17}	b_{16}
HLT	0	0	0	0	1
NOP	0	0	0	0	0

- E' da notare che l'istruzione NOP viene codificata ponendo tutti i bit per l'opcode $b_{23}...b_{16}$ a 0.

Istruzioni nella classe β (argomento di tipo prefissato)

- La seconda sottocategoria (contraddistinta da $b_{20} = b_{19} = 1$) contiene le istruzioni di complemento dei flag, che in realtà dovrebbero appartenere alla categoria di “Salto”, in quanto hanno una funzione relativa ai flag.

	b_{20}	b_{19}	b_{18}	b_{17}	b_{16}
CMC	1	1	0	0	0
CMO	1	1	0	0	1
CMS	1	1	0	1	0
CME	1	1	0	1	1
CMZ	1	1	1	0	0

Istruzioni nella classe β (argomento di tipo prefissato)

- Il motivo per il quale queste istruzioni appartengono a questa categoria è dato dal fatto che i bit utili alla codifica delle istruzioni di “Salto” risultavano insufficienti.
- Questa insufficienza impediva di dare alla codifica delle istruzioni di salto una struttura logica. Per questo si è preferito spostare questo genere di istruzioni in una categoria relativamente più vuota rispetto a quella dei salti

Eniac

- Emulatore di una CPU virtuale progettata e realizzata dal Dott. Mauro Codella e Dott. Dario Dussoni nell'ambito delle loro Tesi di Laurea con il Prof. Enrico Nardelli.

- Attualmente il software è reperibile alla seguente URL :

  http://sourceforge.net/projects/eniac/