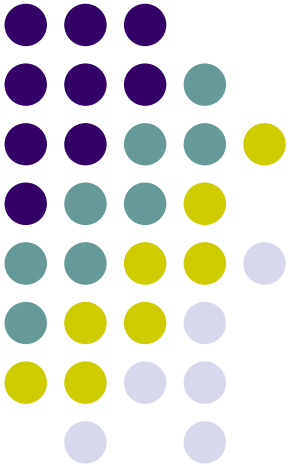
JPD132

Microprocessore e assembly



Il processore JPD132

Il **JPD132** è un ipotetico processore che estende le possibilità del PD32 principalmente con un linguaggio assembly comprendente nuovi metodi di indirizzamento, nuove direttive, nuove istruzioni…

La struttura interna è quasi identica a quella del PD32.

Caratteristiche del JPD132

- modulo di controllo
- registri interni (PC, SR, IR, TEMP1, TEMP2)
- modulo ALU
- 8 registri associati all'ALU (R0, R1, ..., R7)
- bus interno a 32 bit

- interrupt vettorizzato
- convenzione little-endian

Altro hardware

Il calcolatore gestito dal JPD132 è inoltre dotato di:

- memoria centrale (RAM) da 10Kb
- bus di memoria
- bus di I/O
- 7 periferiche di I/O

L'assembly JPD132

L'assembly utilizzabile per il JPD132 è completamente compatibile con l'assembly PD32. Un paio di dettagli:

- i numeri esadecimali non devono necessariamente iniziare con una cifra
- gli indirizzi delle periferiche sono limitati al range 0-63, quindi i programmi possono essere memorizzati a partire dalla locazione di memoria 100H (invece che 400H)

Nuovi metodi di indirizzamento

- assoluto indiretto ..... `MOVL @604,R0`
- auto-incrementante indiretto ..... `MOVL @(R3)+,R0`
- auto-decrementante indiretto ..... `MOVL @-(R3),R0`
- indiretto con registro indice
  pre-indexed ..... `MOVL @24(R2),R0`
- indiretto con registro indice
  post-indexed ..... `MOVL [@24](R2),R0`

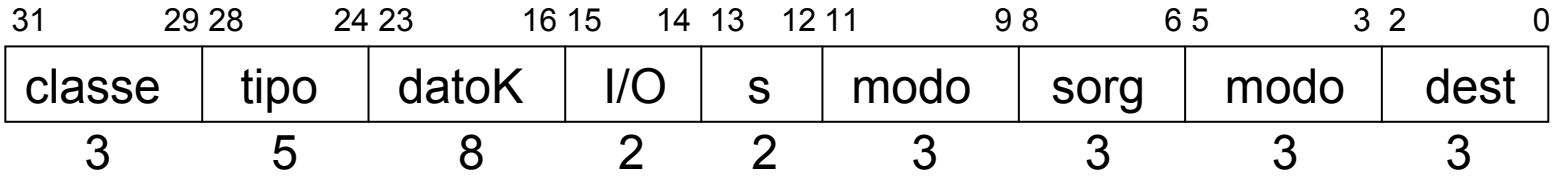
Nuovi metodi di indirizzamento

In alcune istruzioni del JPD132 sono disponibili indirizzamenti non concessi nell'assembly PD32. In pratica si è tentato di permettere più metodi di indirizzamento possibili.

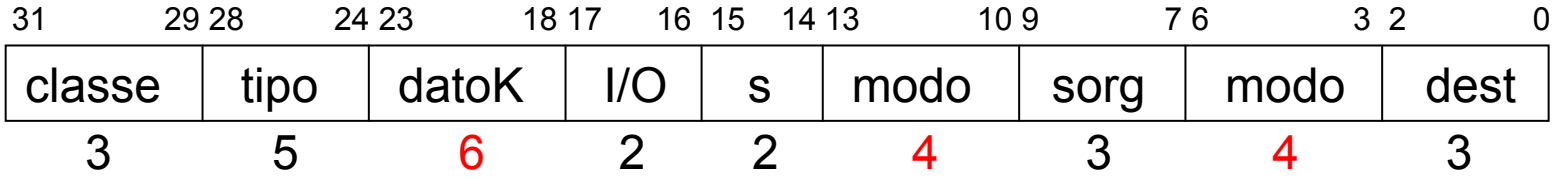
Es: `ADDW 12,@23`

Codifica di istruzioni

PD32



JPD132



Nuove direttive: .DSz

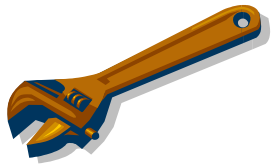
Le direttive `.DSB, .DSW, .DSL` allocano un dato numero di locazioni di memoria. La sintassi è la seguente:

`.DSz` ***espressione***

dove per *espressione* si intende una qualunque espressione aritmetica comprendente gli operatori +, -, *, /, (,) al cui interno si possono utilizzare valori costanti o costanti.

Esempi

```
.DSW 5
.DSL 5*3
.DSL CONST
.DSB CONST-9
.DSL 12*(CONST1-3)*(CONST2+8)
```



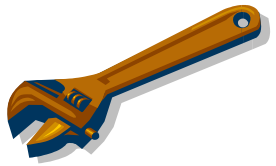
Nuove direttive: .DCz

Le direttive `.DCB`, `.DCW`, `.DCL` allocano in memoria risultati di espressioni aritmetiche. La sintassi è la seguente:

`.DCz` ***espressione[,espressione]***

Possono essere scritte molte espressioni (separate da virgole).

Esempi



```
.DCL 5
.DCW 5,6,2+4,98/2
.DCB CONST,34,1,3*CONST*9
.DCL 12*(CONST-3),3*(3+CONST)
```

Note

- nelle nuove direttive di allocazione di memoria per specificare il valore di una costante si usa direttamente la label (senza '#').
- l'uso di label di variabili in queste espressioni non è consentito sebbene non venga considerato come errore dall'assemblatore.
- l'operando / effettua la divisione intera, quindi:

$$12/2 = 13/2 = 6$$

Nuove istruzioni: EXGz

Le istruzioni **EXGB, EXGW, EXGL** muovono un dato a nella locazione di un dato b e viceversa. "Scambiano di posto" due informazioni. I dati possono essere memorizzati indifferentemente in memoria o nei registri R0,...,R7.

Sintassi: **EXGz S,D**

(S e D non possono essere indirizzati con l'indirizzamento immediato)

Es: **EXGL R5,R2**

Nuove istruzioni: SMULz

Le istruzioni **`SMULB, SMULW, SMULL`** effettuano l'operazione di moltiplicazione segnata.

Sintassi: **`SMULz S,D`**

(D non può essere indirizzato con l'indirizzamento immediato)

Es: **`SMULW #5,312h`**

Nuove istruzioni

Similmente a **`SMULz`** il JPD132 ammette le istruzioni:

- **`UMULz`** (moltiplicazione non segnata)
- **`SDIVz`** (divisione segnata)
- **`UDIVz`** (divisione non segnata)

La sintassi e gli indirizzamenti concessi sono simili a **`SMULz`**.

Nuove istruzioni: NANDz

Il JPD132 estende le istruzioni di tipo logico (classe 3) con:

- **`NANDz S,D`**
- **`NORz S,D`**

In entrambi i casi D non può essere indirizzato con l'indirizzamento immediato.

Estensione dell' istruzione JSR

L'istruzione di chiamata a subroutine può avere due forme:

1) **`JSR label`** (quella usuale del PD32)
2) **`JSR Rx,label`**

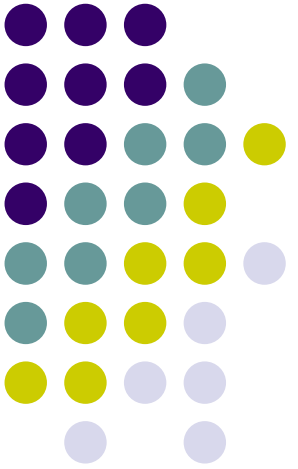
La seconda chiamata memorizza nel registro Rx il contenuto del PC invece che fare una push nello stack. Il ritorno da subroutine in non può avvenire tramite l'istruzione RET, ma deve avvenire con una **`JMP`**.

Esempio

```
      ORG 100H
OP1   DL 5
OP2   DL -3
RIS   DL 0
      CODE

      MOVL OP1,MDCS
      MOVL OP2,MDCS+4
      JSR R1,MD

MDCS: .DSL 1
      .DSL 1
      .DSL 1
      MOVL MDCS+8,RIS
      HALT

MD:   MOVL 4(R1),R0
      SUBL (R1),R0
      JNN POS
      NEGL R0,R0
POS:  MOVL R0,8(R1)
      JMP 12(R1)

      END
```

IL SIMULATORE JPD132

v.1.0 beta



Caratteristiche simulatore

- L'applicazione di simulazione è scritta in Java
- Necessita una Java Virtual Machine v.1.3.1 o superiore
- È multipiattaforma

  (testata in ambiente Windows)
- È distribuita secondo la licenza GPL: http://www.gnu.org/copyleft/gpl.html
- Deve essere posizionata in una cartella con i permessi di scrittura (per il file di configurazione)

Editor

- indentazione automatica
- gestione di diversi file contemporaneamente
- possibilità di cambiare font, dimensione e stile del carattere
- possibilità di specificare una directory di default dove salvare i file
- possibilità di specificare del codice sorgente da caricare in ogni nuova finestra dell'editor

Editor

- comandi per rendere maiuscolo/minuscolo il codice sorgente (ma non i commenti)
- convertitore decimale/esadecimale/binario/ottale/codice ASCII
- commento/decommento automatico del sorgente

Simulatore

- modalità di esecuzione "tutto il programma" (con regolatore di velocità di calcolo)
- modalità di esecuzione "un'istruzione alla volta"
- possibilità di modificare direttamente i valori dei registri interni al processore, le singole celle di memoria centrale, lo Status register, il Program Counter

Simulatore

- possibilità di salvare il contenuto della memoria centrale su file per poi ricaricarlo in una successiva simulazione (opzione utile se si vogliono testare diversi programmi sugli stessi dati)
- indicatore di stima di efficienza (più il valore è alto, più tempo è necessario ad eseguire il programma). Il pulsante "Azz.effic." imposta a 0 il valore dell'indicatore

Simulatore

- pulsante "Random.": setta ad 1 il flip-flop status di tutte le periferiche di input inserendo nel data buffer un valore casuale.

Download

Il simulatore e le sue specifiche sono disponibili all'indirizzo:

http://www.lorenzoblanco.it/jpd132/