Contenuto della lezione

- Come scrivere un programma IJVM
  - Costanti
  - Metodo MAIN
  - Metodi
  - Variabili locali
  - Etichette
  - Commenti
  - I/O
- Alcuni esempi

Costanti globali

- Sono dichiarate in una sezione all'inizio del file
- Il formato può essere decimale (senza prefissi) , esadecimale (se preceduto da "0x") oppure ottale (se preceduto da "o").
- Le costanti possono poi essere usate facendo riferimento al loro nome
- I valori delle costanti sono caricati nella CONSTANT POOL

Sintassi

.constant
    costante1 valore1
    costante2 valore2
    …..
.end-constant

ESEMPIO

.constant
MAX_LENGTH 12
.end-constant
….
LDC_W MAX_LENGTH
…..

Metodo Main

- Ogni programma IJVM deve contenere uno speciale metodo chiamato *main*
- Il *main* deve essere dichiarato prima di ogni altro metodo del programma

Sintassi

.main

….. Dichiarazione delle variabili locali

….. Codice del main

.end-main

Metodi

- I parametri di un metodo sono dichiarati tra parentesi tonde e separati da virgole
- Un metodo può avere zero parametri
- La chiamata di un metodo da parte di codice IJVM deve avvenire seguendo un preciso protocollo:

Metodi

- Push di OBJREF. In JVM è il riferimento all'oggetto sul quale invocare il metodo. IJVM mantiene questa caratteristica per uniformità ma il valore di OBJREF è inessenziale. In ogni caso, un qualsiasi valore per OBJREF deve essere messo in cima allo stack.
- Push dei parametri nell'ordine in cui i parametri di un metodo sono specificati nella dichiarazione.
- Chiamata del metodo con INVOKEVIRTUAL method_name.
- Il valore di ritorno si trova in cima allo stack. L'istruzione IRETURN copia la cima dello stack del metodo sulla cima dello stack di chi ha invocato il metodo.

Sintassi

.method nome_metodo(param1, param2,...)

….. Dichiarazione delle variabili locali

….. Codice del metodo

.end-method

Variabili locali

- Sono dichiarate all'interno di una dichiarazione di metodo e/o del main
- Si possono accedere tramite il riferimento al nome solo all'interno del metodo dove si trova la loro dichiarazione
- Non ci sono variabili globali

Sintassi

.var

    var1

    var2

    …..

.end-var

ESEMPIO

….

ISTORE var1

…..

Etichette

- Servono ad identificare istruzioni in un programma che possono essere l'argomento di un'istruzione di salto
- Le etichette sono accessibili solo all'interno del metodo dove sono state dichiarate (I salti possono avvenire all'interno di un metodo e non "tra metodi")
- Le etichette sono stringhe alfanumeriche terminate dal carattere ":"

Sintassi

```
ILOAD total
LDC_W max
ISUB
IFLT lt_max    // If total < max, goto lt_max
GOTO gte_max  //  else total >= max, goto gte_max
lt_max: HALT
gte_max:  // Possono essere su linee separate
                INVOKEVIRTUAL error
```

Commenti

- Servono ad inserire annotazioni al codice prodotto (FONDAMENTALI)
- Si indica l'inizio di un commento con due slash "//"
- Tutti i caratteri dal "//" alla fine della linea sono trattati come commenti

Struttura generale di un programma IJVM

.constant
    costante1 valore1
    costante2 valore2
    …..
.end-constant
.main
….. Dichiarazione delle variabili locali del main
….. Codice del main
.end-main
.method nome_metodo1(param1, param2,...)
….. Dichiarazione delle variabili locali del metodo1
….. Codice del metodo1
.end-method
.method nome_metodo2(param1, param2,...)
….. Dichiarazione delle variabili locali del metodo2
….. Codice del metodo2
.end-method
….
….

Problema numero 1

- Scrivere un programma IJVM che scambia il contenuto di due variabili locali X e Y

```
.main

.var
X
Y
.end-var

start:  BIPUSH 0x10
        ISTORE X      // inizializzo X=0x10
        BIPUSH 0x20
        ISTORE Y      // inizializzo Y=0x20
//      scambia il contenuto
        ILOAD X
        ILOAD Y
        ISTORE X
        ISTORE Y
        HALT
.end-main
```

Problema numero 2

- Scrivere un programma IJVM che esegua X=min(X,Y) e Y=max(X,Y)

```
.constant
tre 0x03
quattro 0x04
.end-constant

.main

.var
X
Y
.end-var

start:
        LDC_W  quattro
        ISTORE X
        LDC_W  tre
        ISTORE Y
        ILOAD X
        ILOAD Y
        ISUB
        IFLT end // if X<Y vai a end
        ILOAD X
        ILOAD Y
        ISTORE X
        ISTORE Y
end:    HALT
.end-main
```

Problema numero 3

- Scrivere un programma IJVM che sommi i primi 32 numeri interi (struttura ciclo **do-while**)

```
.main

.var
sum
cont
.end-var

START:
        BIPUSH 0x0
        ISTORE sum    // inizializzo sum=0x00
        BIPUSH 0x0
        ISTORE cont   // inizializzo cont=0x00
CICLO:  IINC cont 1   // incrementa contatore
        ILOAD cont    // carica cont sullo stack
        ILOAD sum     // carica sum  sullo stack
        IADD          // sum+cont
        ISTORE sum    // memorizza somma
        ILOAD cont    // carica  cont sullo stack
        BIPUSH 0x20   // carica la costante 32
        IF_ICMPEQ  L2 // se cont==32 vai a L2
        GOTO CICLO    // salta all'inizio del ciclo
L2:     HALT
.end-main
```

Problema numero 4

- Scrivere un programma IJVM che sommi i primi 32 numeri interi (struttura ciclo **while**)

```
.main
.var
sum
cont
.end-var

START:  BIPUSH 0x0
        ISTORE sum     // inizializzo sum=0x00
        BIPUSH 0x0
        ISTORE cont    // inizializzo cont=0x00
CICLO:  IINC cont 1    // incrementa contatore
        ILOAD cont     // carica cont sullo stack
        BIPUSH 0x20    // carica la costante 32
        ISUB           // cont-32
        IFLT LL        // se cont-32<0 vai a LL
        ILOAD cont     // carica cont sullo stack
        BIPUSH 0x20    // carica la costante 32
        ISUB           // cont-32
        IFEQ LL        // se cont-32=0 vai a LL
        GOTO FINE      // vai a FINE (cont>32)
LL:     ILOAD sum      // carica sum  sullo stack
        ILOAD cont     // carica cont sullo stack
        IADD           // sum+cont
        ISTORE sum     // memorizza somma
        GOTO CICLO     // salta all'inizio del ciclo
FINE:   HALT
.end-main
```

Problema numero 5

- Scrivere un programma IJVM che conti il numero di bit a 1 in una parola X di 32 bit

```
.constant
VALX  0xFA10
.end-constant

.main
.var
X    // parola di memoria da analizzare
C    // contatore di bit uguali a 1
.end-var
START:  LDC_W  VALX  // init. X con il valore costante VALX
        ISTORE X
        BIPUSH 0x0
        ISTORE C      // inizializzo C=0x0
CICLO:  ILOAD X       // carica X sullo stack
        IFEQ FINE     // se X=0 vai a FINE (nessun bit a 1)
        ILOAD X
        IFLT NEG      // se X<0 vai a NEG (MSB = 1)
        GOTO L1
NEG:    IINC C 1      // incrementa C (perché c'e' un 1)
L1:     ILOAD X       // X = X + X (ovvero left-shift X)
        DUP
        IADD
        ISTORE X
        GOTO CICLO
FINE:   HALT
.end-main
```