

# IJVM

## Esercizio 1

- (a) Si programmi il metodo in IJVM/JAS che restituisce il valore assoluto di un intero ( $\text{abs}(n)$ )
- (b) Si scriva il programma IJVM/JAS corrispondente al seguente frammento di codice C:  

```
int y = 12;  
int x = 10-K;  
int y = abs(x) + 2 - y;
```

dove K è una costante definita nel programma

## Esercizio 2

- (a) Si scriva il programma IJVM/JAS corrispondente al seguente frammento di codice C:  

```
int x = -3-2*K;  
int y = 23 + f(2, x+1) - K;
```

dove: (1) K è una costante definita nel programma; (2) f è un metodo a due argomenti (da non definire)
- (b) Mostrare come cambia lo stack degli operandi durante l'esecuzione della prima istruzione (assumere che il valore di K sia 4).

## Esercizio 3

- (a) Programmare il metodo in IJVM/JAS che realizza la seguente funzione C:  

```
int f(int x, int y) {  
    int z = x+y - 2;  
    return z + z;  
}
```
- (b) Scrivere il codice IJVM/JAS corrispondente al seguente frammento di codice C  

```
int y = N+2;  
int x = f(y-1, y) +10;
```

dove N è una costante definita nel programma
- (c) Descrivere come sono realizzate su MIC-1 le chiamate a metodo IJVM. Spiegare le fasi di prologo e di epilogo delle chiamate a procedura, ed indicare come sono realizzate in IJVM su MIC-1
- (d) Illustrare come evolve lo stack nell'esecuzione della seconda istruzione ( $\text{int } x = f(y-1, y) + 10$ ), compresa l'esecuzione della chiamata al metodo f. Si assuma che N valga 5.
- (e) Descrivere come è composto il record di attivazione della chiamata al metodo f.

## Esercizio 4

- (a) Programmare in IJVM/JAS il metodo che realizza la seguente funzione C:  

```
int modulo(int dividendo, int divisore) {  
    int quoziente = 0; int resto = dividendo;  
    while (divisore <= resto) { resto = resto - divisore; quoziente = quoziente + 1; }  
    return resto;  
}
```
- (b) Scrivere il codice IJVM/JAS che valuti la seguente espressione:  $C - \text{modulo}(27, x+2) + 21$  (dove C è una costante ed x una variabile definite nel programma)

## Esercizio 5

- (a) Trasformare in notazione polacca inversa l'espressione:  $[3 + (z-5) * 4 - y] * 2$
- (b) Codificare in IJVM/JAS il seguente frammento di codice C, dove C è una costante il cui valore è 8:  

```
z = 1;  
y = 2;  
z = [3 + (z-5) * 4 - y] * 2;  
if (z >= 0) {  
    y = y-C;  
} else {  
    y = y+3;
```

### Esercizio 6

Dato il seguente frammento di codice IJVM/JAS

i01: ILOAD x i02: LDC_W K1 i03: IADD i04: BIPUSH 25 i05: ISUB i06: DUP i07: IFLT i11 i08: POP i09: IINC x -1 i10: GOTO i12 i11: ISTORE x i12: ...	(a) Illustrare come cambia lo stack degli operandi e la variabile x durante la sua esecuzione (indicare le transizioni con le etichette che identificano le diverse istruzioni). Si supponga che la variabile x inizialmente (prima di eseguire i01) contenga il valore 9 e che K1 sia una variabile di valore 10. (b) Tradurre il frammento di codice in istruzioni C-like
--	--

### Esercizio 7

Dato il seguente frammento di codice IJVM/JAS

label1: ILOAD x label2: ILOAD y label3: ISUB label4: BIPUSH 2 label5: IF_ICMPEQ label7 label6: GOTO label9 label7: BIPUSH 20 label8: DUP label9: ...	(a) Illustrare come cambia lo stack (degli operandi) durante la sua esecuzione (indicare con le etichette che identificano le diverse istruzioni le transizioni dello stack). Si supponga che la variabile x abbia valore 10 e y abbia valore 12. (b) Mostrare la codifica esadecimale delle istruzioni con etichetta label1 e label6, ipotizzando che la variabile x si trovi all'indirizzo LV+4 nel record di attivazione.
--	---

### Esercizio 8

- Si converta la seguente espressione da notazione polacca inversa a notazione infissa:  $3\ 5 - 4 + 2 *$
- Mostrare come l'espressione sia valutata tramite lo stack degli operandi

### Esercizio 9

- Programmare il metodo in IJVM/JAS che realizza la seguente funzione C:  

```
int g(int x) { int y;  
    if (x>=0) {y = x+1;} else {y=x-1;}  
    return y*2;}
```
- Scrivere il codice IJVM/JAS corrispondente al seguente frammento di codice C:  

```
int m = 5;  
int n = g(m+1) - K; // dove K è una costante definita nel programma
```
- Descrivere il record di attivazione corrispondente all'invocazione della funzione g(m+1)

### Esercizio 10

- Programmare in IJVM/JAS il metodo che realizza la funzione del calcolo del MCD secondo l'algoritmo di Euclide:  

```
int mcd(int x, int y) { //assumere che x e y siano numeri positivi 0  
    while (x != y) {if (x>y) {x = x-y;} else {y = y-x;}}  
    return x;}
```
- Scrivere il codice IJVM/JAS corrispondente alla seguente assegnazione C, dove n e m sono variabili e K una costante:  

```
n = - mcd(m-4, m+K);
```
- Descrivere il record di attivazione corrispondente all'invocazione della funzione mcd(16, 24)

### Esercizio 11

Scrivere il codice IJVM/JAS che calcoli la seguente funzione ricorsiva f:  
`int f(int x, int y) { if (x<0) {return y;} else {return f(x-1, y)-2;}}`

### Esercizio 12

Dato il seguente codice IJVM/JAS

<pre>.method dispari(x) .var .end-var     <b>ILOAD x (*)</b>     BIPUSH 1     IAND     IRETURN .end-method .constant     OBJREF 0x40     K 21 .end-constant</pre>	<pre>.main .var     x     y .end-var     BIPUSH 5     LDC_W K     IADD     ISTORE x     LDC_W OBJREF     ILOAD x     INVOKEVIRTUAL dispari</pre>	<pre><b>IFEQ then (*)</b>     BIPUSH 0     ISTORE y     GOTO fine then:    LDC_W K     <b>DUP (*)</b>     IADD     <b>BIPUSH 1 (*)</b>     ISUB     ISTORE y fine:    HALT .end-main</pre>
---	--	--

(a) Scrivere e spiegare il formato binario/esadecimale delle istruzioni contrassegnate da (*)	
(b) Descrivere il record di attivazione della chiamata al metodo dispari	SP-->
(c) Illustrate l'evoluzione dello stack durante l'esecuzione della invocazione del main, partendo dalla seguente configurazione iniziale dell'emulatore IJVM	(y) 0
	LV--> (x) 0

### Esercizio 13

Dato il seguente codice IJVM/JAS

<pre>.method max (p1, p2)     ILOAD p1     <b>ILOAD p2 (*)</b>     ISUB     <b>IFLT maxP2 (*)</b>     ILOAD p1     IRETURN maxP2: ILOAD p2     <b>IRETURN (*)</b> .end-method</pre>	<pre>.constant     OBJREF 0x40     K 10 .end-constant  .main .var     n     m .end-var</pre>	<pre>BIPUSH 13 ISTORE n BIPUSH 10 ILOAD n ISUB ISTORE m LDC_W OBJREF ILOAD m LDC_W K DUP</pre>	<pre>IADD IADD ILOAD n BIPUSH 5 ISUB INVOKEVIRTUAL max ISTORE n <b>IINC m 1 (*)</b> .end-main</pre>						
<p>(a) Riprodurre il codice IJVM/JAS in (pseudo-)codice C</p> <p>(b) Descrivere il formato esadecimale (codice operativo + operandi) delle istruzioni contrassegnate da (*)</p> <p>(c) Illustrate l'evoluzione dello stack durante l'esecuzione del main, partendo dalla configurazione iniziale dell'emulatore IJVM riprodotta qui di lato, indicando sullo stack anche i record di attivazione e le istruzioni del metodo max</p>			<table><tr><td>SP--&gt;</td><td>"PC fine"</td></tr><tr><td>(m)</td><td>0</td></tr><tr><td>LV--&gt; (n)</td><td>0</td></tr></table>	SP-->	"PC fine"	(m)	0	LV--> (n)	0
SP-->	"PC fine"								
(m)	0								
LV--> (n)	0								

**Esercizio 14**

Dato il seguente codice IJVM/ JAS

<pre>.method f (p1)     ILOAD p1     IFLT risZero     ILOAD p1     IFEQ risZero     ILOAD p1     BIPUSH 1     ISUB     IRETURN risZero: BIPUSH 0     IRETURN .end-method</pre>	<pre>.constant     OBJREF 0x40     K 42 .end-constant  .main .var     x     y .end-var</pre>	<pre>BIPUSH 1 ISTORE x LDC_W OBJREF LDC_W OBJREF ILOAD x INVOKEVIRTUAL f INVOKEVIRTUAL f LDC_W K IADD ISTORE y .end-main</pre>	.
(d) Riscrivere in pseudo-codice C il metodo f ed il main			SP-->
(e) Illustrate l'evoluzione dello stack durante l'esecuzione del main, partendo dalla configurazione iniziale dell'emulatore IJVM riprodotta qui di lato, indicando sullo stack anche i record di attivazione e l'esecuzione delle istruzioni del metodo f			(y)
			LV-->
			(x)
			"PC fine"
			0
			0