**Esercizio 1:** Tradurre il seguente codice C in assembly LEGv8.

void swap(long long int\* x, long long int\* y)

{

 long long int t;

 t = \*y;

 \*y = \*x;

 \*x = t;

}

long long int median3(long long int a, long long int b, long long int c)

{

 if (a > c)

 swap(&a, &c);

 if (a > b)

 swap(&a, &b);

 if (b > c)

 swap(&b, &c);

 return b;

}

Nota: Gli argomenti vanno passati a swap tramite i registri x0, x1 e a median3 tramite i registri x0, x1, x2 e il risultato va restituito tramite x0. In median3 assumere che swap possa modificare qualunque “registro non preservato tra chiamate”.

**Esercizio 2:** Tradurre il seguente codice C in assembly LEGv8.

void FIR\_filter(long long int h[], long long int N, long long int x[],

long long int LEN, long long int y[])

{

 long long int n;

 long long int i;

 long long int S;

 for (n = N; n < LEN; n++) {

 S = 0;

 for (i = 0; i <= N; i++)

 S += h[i] \* x[n - i];

 y[n - N] = S;

 }

}

// Suggerimento: i parametri vengono passati in X0, ..., X4

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> n, e le operazioni principali, e.g. // S = 0

**Esercizio 3:** Tradurre il seguente codice C in assembly LEGv8.

void autocorrelation(long long int x[], long long int y[], long long int LEN)

{

 long long int n;

 long long int m;

 long long int A;

 for (n = 0; n < LEN; n++) {

 A = 0;

 for (m = 0; m < LEN - n; m++)

 A += x[m] \* x[m + n];

 y[n] = A;

 }

}

// Suggerimento: i parametri vengono passati in X0, X1, X2

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> n, e le operazioni principali, e.g. // S = 0

**Esercizio 4:** Tradurre il seguente codice C in assembly LEGv8.

void updateX(long long int x, long long int X[], long long int N);

long long int FIR(long long int x, long long int h[], long long int X[],

 long long int N)

{

 long long int i;

 long long int S;

 updateX(x, X, N);

 S = 0;

 for (i = 0; i < N; i++)

 S += h[i] \* X[i];

 return S;

}

// Suggerimento: i parametri di updateX vengono passati in X0, X1, X2;

// i parametri di FIR vengono passati in X0, X1, X2, X3 e il risultato in X0.

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni principali, e.g. // S = 0

**Esercizio 4bis:** Tradurre il seguente codice C in assembly LEGv8.

void updateX\_D(double x, double X[], long long int N)

{

 for (size\_t i = N - 1; i > 0; i--)

 X[i] = X[i - 1];

 X[0] = x;

}

double FIR\_D(double x, double h[], double X[], long long int N)

{

 double S;

 updateX\_D(x, X, N);

 S = 0;

 for (size\_t i = 0; i < N; i++)

 S += h[i] \* X[i];

 return S;

}

// Suggerimento: i parametri di updateX vengono passati in D0, X0, X1;

// i parametri di FIR vengono passati in D0, X0, X1, X2, e il risultato in D0.

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni principali, e.g. // S = 0

**Esercizio 5:** Tradurre il seguente codice C in assembly LEGv8.

void IIR2\_filter(long long int x[], long long int c[], long long int LEN,

long long int y[])

{

 long long int i;

 long long int A;

 long long int z[2];

 z[0] = 0;

 z[1] = 0;

 for (i = 0; i < LEN; i++) {

 A = x[i] - z[0] \* c[1] - z[1] \* c[2];

 y[i] = c[3] \* A + c[4] \* z[0] + c[5] \* z[1];

 z[1] = z[0];

 z[0] = A;

 }

}

// Suggerimento: i parametri di IIR2\_filter vengono passati in X0, X1, X2, x3

// e z[] va salvato nello stack.

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni principali, e.g. // Z[1] = 0

**Esercizio 6:** Tradurre il seguente codice C in assembly LEGv8.

long long int LatcAPFilt(long long int x, long long int K[], long long int G[],

long long int N)

{

 long long int F;

 long long int P;

 long long int i;

 F = x;

 for (i = N - 1; i >= 0; i--)

 {

 P = (F - G[i]) \* K[i];

 F = F + P;

 G[i + 1] = P + G[i];

 }

 G[0] = F;

 return G[N];

}

**Esercizio 7:** Tradurre il seguente codice C in assembly LEGv8.

long long int med3(long long int a, long long int b, long long int c);

void medfilter(long long int x[], long long int y[], long long int Len)

{

 long long int n;

 for (n = 0; n < Len - 2; n++)

 y[n] = med3(x[n], x[n + 1], x[n + 2]);

}

// Suggerimento: i parametri di medfilter vengono passati in X0, X1, X2;

// i parametri di med3 vengono passati in X0, X1, X2;

// il risultato viene restituito in X0.

// Usare il più possibile i registri per le variabili locali.

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> n, e le operazioni principali, e.g. // n = 0

**Esercizio 8:** Tradurre il seguente codice C in assembly LEGv8.

double FIR1\_D(double h[], double X[], long long int N);

double LMS1(double d, double mu, double h[], double X[], long long int N)

{

 long long int i;

 double y, e, em;

 y = FIR1\_D(h, X, N);

 e = d - y;

 em = e \* mu;

 for (i = 0; i < N; i++)

 h[i] += em \* X[i];

 return y;

}

// Suggerimento: i parametri vengono passati alle funzioni nei registri X o D a

// seconda della tipologia del dato sempre a partire dal registro 0.

// Il risultato viene restituito nel registro 0.

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni principali, e.g. // e = d – y;

**Esercizio 9:** Tradurre il seguente codice C in assembly LEGv8.

void crosscorrelation(float x[], float y[], float z[], long long int LEN)

{

 long long int n;

 long long int m;

 float A;

 for (n = 0; n < LEN; n++) {

 A = 0;

 for (m = 0; m < LEN - n; m++)

 A += x[m] \* y[m + n];

 z[n] = A;

 }

}

// I parametri vengono passati alle funzioni a partire dal registro 0.

// Fare molta attenzione al formato dei dati!

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> n, e le operazioni principali, e.g. // A = 0;

**Esercizio 10:** Tradurre il seguente codice C in assembly LEGv8.

float mu(void);

float denominator(float x[], long long int LEN)

{

 long long int n;

 float den=0;

 for (n = 0; n < LEN; n++)

 den += x[n] \* x[n];

 return ( mu() / den);

}

// I parametri vengono passati alle funzioni a partire dal registro 0.

// I risultati vengono restituiti nel registro 0 del tipo opportuno.

// Fare molta attenzione al formato dei dati!

// La funzione mu() potrebbe modificare qualsiasi registro che non sia un “saved

// register”.

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> n, e le operazioni principali, e.g. // den=0;