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

```c
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.

```c
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)

{

	for (size_t i = N - 1; i > 0; i--)
		X[i] = X[i - 1];
	X[0] = x;
}


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.

```c
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.

```c
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.

```c
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.

```c
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;
```

Esercizio 11: Tradurre il seguente codice C in assembly LEGv8.

```
float autocor1(float x[], long long int L, long long int n);

void autocorrel(float x[], float y[], long long int LEN)
{
	long long int n;
	for (n = 0; n < LEN; n++) {
		y[n] = autocor1(x, LEN - n, n);
	}
}

// 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 tipo dei dati passati alle funzioni!
// La funzione autocor1() 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. // LEN - n;
```

Esercizio 12: Tradurre il seguente codice C in assembly LEGv8.

```
float LatcAPFiltFloat(float x, float K[], float G[], long long int N)
{
	float F;
	float 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];
}
// I parametri vengono passati alla funzione nei registri opportuni a partire dal
// registro 0.
// Il risultato viene restituito nel registro 0 del tipo opportuno.
// Fare molta attenzione al tipo dei dati!
// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni
principali, e.g. // F-G[i]
```

Esercizio 13: Tradurre il seguente codice C in assembly LEGv8.

```
double Mean2D(double A[8][8])
{
	long long int i, j;
	double M;

	M = 0.0;
	for (i = 0; i < 8; i++)
		for (j = 0; j < 8; j++)
			M+= A[i][j];
	return M / 64.0;
}

// Suggerimenti:
// Seguire le regole LEGv8 standard per il passaggio dei parametri della funzione
// Per i registri floating-point: i primi 8 registri sono usati per gli argomenti e i
risultati e non sono “preserved across a call”, i secondi 8 sono dei registri
“preserved across a call”, gli ultimi 16 non sono “preserved across a call”.
// Le parole binarie che rappresentano le costanti 0.0 e 64.0 vanno preparate nei
// registri X e trasferite nei registri D tramite lo stack.
// La rappresentazione binaria del double 64.0 può essere ottenuta come 1029<<52
// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni
principali, e.g. // M = 0.0
```

Esercizio 14: Tradurre il seguente codice C in assembly LEGv8.

```c
double myexp(double C);

void exp2D(double A[1024][1024])
{
	long long int i, j;

	for (i = 0; i < 1024; i++)
		for (j = 0; j < 1024; j++)
			A[i][j] = myexp(A[i][j]);

}
// Suggerimenti:
// Seguire le regole LEGv8 standard per il passaggio dei parametri della funzione
// Assumere che myexp possa sporcare qualunque registro che non sia “preserved across a call”. Per i dati di exp2D,conviene usare il più possibile i registri “preserved across a call”.
// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i,
```

Esercizio 15: Tradurre il seguente codice C in assembly LEGv8.

```c
void FIR_s(double h[], long long int N, double x[], long long int LEN,
		double y[])
{
	long long int n;
	long long int i;
	double S;
	long long int K = 2 * N - 1;

	for (n = N; n < LEN; n++) {
		S = 0.0;
		for (i = 0; i <= N; i++)
			S += h[i] * x[n-K+i];
		y[n - N] = S;
	}
}

// Suggerimenti:
// Seguire le regole LEGv8 standard per il passaggio dei parametri della funzione
// La parola binaria che rappresenta la costante 0.0 va preparata nei
// registri X e trasferita nei registri D tramite lo stack.

// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni principali, e.g. // S = 0.0
```

Esercizio 16: Tradurre il seguente codice C in assembly LEGv8.

```c
void ReadCoeffPoly1(float a[], long long int N);
float Poly1(float x, float *a, long long int N)
{
	long long int i;
	float y;

	ReadCoeffPoly1(a, N);
	y = a[N];
	for (i = N - 1; i >= 0; i--) {
		y *= x;
		y += a[i];
	}
	return y;
}

// 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 ReadCoeffPoly1 potrebbe modificare qualsiasi registro che non sia
// un “saved register”.
// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni
principali, e.g. // y = a[N];
```

Esercizio 17: Tradurre il seguente codice C in assembly LEGv8.

```c
double myPow(double x, long long int i);
double Poly2(double x, double a[], long long int N)
{
	long long int i;
	double y;

	y = a[0];
	for (i = 1; i <= N; i++) {
		y += a[i] * myPow(x, i);
	}
	return y;
}

// 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 myPow potrebbe modificare qualsiasi registro che non sia un “saved
// register”.
// Suggerimento: usare i “saved register” il più possibile.
// I saved register floating point sono quelli dall’8 al 16.
// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le
operazioni principali, e.g. // y = a[0];
```

Esercizio 18: Tradurre il seguente codice C in assembly LEGv8.

```
double zero(void);

double fun(double x[8][8]) {
	long long int u, v;
	double sum;

	sum = zero();
	for (u = 0; u < 8; u++) {
		for (v = 0; v < 8; v++) {
			sum += x[u][v];
		}
	}
	return sum;
}

// 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 funzioni zero() potrebbe modificare qualsiasi registro che non sia
// un “saved register”.
// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> u, e le
operazioni principali, e.g. // sum = zero();
```

Esercizio 19: Tradurre il seguente codice C in assembly LEGv8.

```
float myrand(long long int *seed);
void funfun(long long int* seed, float A[128][128])
{
	long long int i, j;

	for (i = 0; i < 128; i++)
		for (j = 0; j < 128; j++)
			A[i][j] = myrand(seed);
}
// 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 myrand potrebbe modificare qualsiasi registro che non sia
// un “saved register”.
// Usare il più possibile i “saved register”.
// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni
principali, e.g. // myrand(seed).
```

Esercizio 20: Tradurre il seguente codice C in assembly LEGv8.

```
float f0(float x);
void myAdd(float A[8][8], float B[8][8] )
{
      long long int i, j;

      for (i = 0; i < 8; i++)
            for (j = 0; j < 8; j++)
                  A[i][j] = A[i][j] + f0(B[i][j]);
}
// 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 funzioni f0 () potrebbe modificare qualsiasi registro che non sia
// un “saved register”.
// Usare il più possibile i “saved register”.
// Ho trovato conveniente mettere l’indirizzo di A[i][j] in un saved register.
// Specificare con commenti l'uso fatto dei registri, e.g. X9 -> i, e le operazioni
principali, e.g. // A[i][j] + f0(A[i][j]);
```