```
1  // headers
2  #include <stdio.h>
3
4  #include <cuda.h> // for CUDA
5
6  // global variables
7  int inputLength=5;
8
9  float *hostInput1=NULL;
10 float *hostInput2=NULL;
11 float *hostOutput=NULL;
12
13 float *deviceInput1=NULL;
14 float *deviceInput2=NULL;
15 float *deviceOutput=NULL;
16
17 // global kernel function definition
18 __global__ void vecAdd(float *in1,float *in2,float *out,int len)
19 {
20     // variable declarations
21     int i=blockIdx.x * blockDim.x + threadIdx.x;
22     // code
23     if(i < len)
24     {
25         out[i]=in1[i]+in2[i];
26     }
27 }
28
29 int main(int argc,char *argv[])
30 {
31     // function declarations
32     void cleanup(void);
33
34     // code
35     // allocate host-memory
36     hostInput1=(float *)malloc(inputLength * sizeof(float));
37     if(hostInput1== NULL)
38     {
39         printf("CPU Memory Fatal Error = Can Not Allocate Enough Memory For
           Host Input Array 1.\nExitting ...\n");
40         cleanup();
41         exit(EXIT_FAILURE);
42     }
43
44     hostInput2=(float *)malloc(inputLength * sizeof(float));
45     if(hostInput2== NULL)
46     {
47         printf("CPU Memory Fatal Error = Can Not Allocate Enough Memory For
           Host Input Array 2.\nExitting ...\n");
48         cleanup();
49         exit(EXIT_FAILURE);
50     }
51
52     hostOutput=(float *)malloc(inputLength * sizeof(float));
53     if(hostOutput== NULL)
54     {
```

```
55          printf("CPU Memory Fatal Error = Can Not Allocate Enough Memory For
              Host Output Array.\nExitting ...\n");
56          cleanup();
57          exit(EXIT_FAILURE);
58      }
59
60      // fill above input host vectors with arbitary but hard-coded data
61      hostInput1[0]=101.0;
62      hostInput1[1]=102.0;
63      hostInput1[2]=103.0;
64      hostInput1[3]=104.0;
65      hostInput1[4]=105.0;
66
67      hostInput2[0]=201.0;
68      hostInput2[1]=202.0;
69      hostInput2[2]=203.0;
70      hostInput2[3]=204.0;
71      hostInput2[4]=205.0;
72
73      // allocate device-memory
74      int size=inputLength * sizeof(float);
75      cudaError_t err=cudaSuccess;
76      err=cudaMalloc((void **)&deviceInput1,size);
77      if(err!=cudaSuccess)
78      {
79          printf("GPU Memory Fatal Error = %s In File Name %s At Line No. %d.
              \nExitting ...\n",cudaGetErrorString(err),__FILE__,__LINE__);
80          cleanup();
81          exit(EXIT_FAILURE);
82      }
83
84      err=cudaMalloc((void **)&deviceInput2,size);
85      if(err!=cudaSuccess)
86      {
87          printf("GPU Memory Fatal Error = %s In File Name %s At Line No. %d.
              \nExitting ...\n",cudaGetErrorString(err),__FILE__,__LINE__);
88          cudaFree(deviceInput1);
89          cleanup();
90          exit(EXIT_FAILURE);
91      }
92
93      err=cudaMalloc((void **)&deviceOutput,size);
94      if(err!=cudaSuccess)
95      {
96          printf("GPU Memory Fatal Error = %s In File Name %s At Line No. %d.
              \nExitting ...\n",cudaGetErrorString(err),__FILE__,__LINE__);
97          cleanup();
98          exit(EXIT_FAILURE);
99      }
100
101     // copy host memory contents to device memory
102     err=cudaMemcpy(deviceInput1,hostInput1,size,cudaMemcpyHostToDevice);
103     if(err!=cudaSuccess)
104     {
105         printf("GPU Memory Fatal Error = %s In File Name %s At Line No. %d.
              \nExitting ...\n",cudaGetErrorString(err),__FILE__,__LINE__);
```

```
106         cleanup();
107         exit(EXIT_FAILURE);
108     }
109
110     err=cudaMemcpy(deviceInput2,hostInput2,size,cudaMemcpyHostToDevice);
111     if(err!=cudaSuccess)
112     {
113         printf("GPU Memory Fatal Error = %s In File Name %s At Line No. %d.
              \nExitting ...\n",cudaGetErrorString(err),__FILE__,__LINE__);
114         cleanup();
115         exit(EXIT_FAILURE);
116     }
117
118     // cuda kernel configuration
119     dim3 DimGrid=dim3(ceil(inputLength/256.0),1,1);
120     dim3 DimBlock=dim3(256,1,1);
121     vecAdd<<<DimGrid,DimBlock>>>
          (deviceInput1,deviceInput2,deviceOutput,inputLength);
122
123     // copy device memory to host memory
124     err=cudaMemcpy(hostOutput,deviceOutput,size,cudaMemcpyDeviceToHost);
125     if(err!=cudaSuccess)
126     {
127         printf("GPU Memory Fatal Error = %s In File Name %s At Line No. %d.
              \nExitting ...\n",cudaGetErrorString(err),__FILE__,__LINE__);
128         cleanup();
129         exit(EXIT_FAILURE);
130     }
131
132     // results
133     int i;
134     for(i=0;i<inputLength;i++)
135     {
136         printf("%f + %f = %f\n",hostInput1[i],hostInput2[i],hostOutput[i]);
137     }
138
139     // total cleanup
140     cleanup();
141
142     return(0);
143 }
144
145 void cleanup(void)
146 {
147     // code
148
149     // free allocated device-memory
150     if(deviceInput1)
151     {
152         cudaFree(deviceInput1);
153         deviceInput1=NULL;
154     }
155
156     if(deviceInput2)
157     {
158         cudaFree(deviceInput2);
```

```
159         deviceInput2=NULL;
160     }
161
162     if(deviceOutput)
163     {
164         cudaFree(deviceOutput);
165         deviceOutput=NULL;
166     }
167
168     // free allocated host-memory
169     if(hostInput1)
170     {
171         free(hostInput1);
172         hostInput1=NULL;
173     }
174
175     if(hostInput2)
176     {
177         free(hostInput2);
178         hostInput2=NULL;
179     }
180
181     if(hostOutput)
182     {
183         free(hostOutput);
184         hostOutput=NULL;
185     }
186 }
187
```