ASSIGNMENT - V

```
#include <stdio.h>
#include <stdlib.h>
#include <CL/cl.h>
#include <time.h>

#define N 2000000000  // size of the vectors

const char *kernel_src =
"__kernel void vec_add(__global float* A, __global float* B, __global float* C) {\n"
"    int id = get_global_id(0);\n"
"    C[id] = A[id] + B[id];\n"
"}\n";

int main() {
    size_t size = N * sizeof(float);
    float *A = (float*)malloc(size);
    float *B = (float*)malloc(size);
    float *C = (float*)malloc(size);
    for (int i = 0; i < N; i++) {
        A[i] = (float)(i % 1000);
        B[i] = (float)(i % 500);
    }

    // CPU Vector Addition
    clock_t start = clock();
    for (int i = 0; i < N; i++) {
        C[i] = A[i] + B[i];
    }
    clock_t end = clock();
    printf("CPU Time: %.4f ms\n", (end - start) * 1000.0 / CLOCKS_PER_SEC);

    // OpenCL Initialization
```

```
    cl_platform_id platform_id;
    cl_device_id device_id;
    cl_uint num_platforms, num_devices;
    clGetPlatformIDs(1, &platform_id, &num_platforms);
    clGetDeviceIDs(platform_id, CL_DEVICE_TYPE_GPU, 1, &device_id, &num_devices);

    cl_context context = clCreateContext(NULL, 1, &device_id, NULL, NULL, NULL);
    cl_command_queue queue = clCreateCommandQueue(context, device_id, 0, NULL);

    cl_mem d_A = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, size,
A, NULL);
    cl_mem d_B = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, size,
B, NULL);
    cl_mem d_C = clCreateBuffer(context, CL_MEM_WRITE_ONLY, size, NULL, NULL);

    cl_program program = clCreateProgramWithSource(context, 1, &kernel_src, NULL, NULL);
    clBuildProgram(program, 1, &device_id, NULL, NULL, NULL);
    cl_kernel kernel = clCreateKernel(program, "vec_add", NULL);
    clSetKernelArg(kernel, 0, sizeof(cl_mem), &d_A);
    clSetKernelArg(kernel, 1, sizeof(cl_mem), &d_B);
    clSetKernelArg(kernel, 2, sizeof(cl_mem), &d_C);

    size_t global_work_size = N;

    start = clock();
    clEnqueueNDRangeKernel(queue, kernel, 1, NULL, &global_work_size, NULL, 0, NULL, NULL);
    clFinish(queue);
    end = clock();
    printf("GPU/OpenCL Time: %.4f ms\n", (end - start) * 1000.0 / CLOCKS_PER_SEC);

    clEnqueueReadBuffer(queue, d_C, CL_TRUE, 0, size, C, 0, NULL, NULL);

    // Optional validation
    for (int i = 0; i < 10; i++) {
```

```
        printf("C[%d] = %f\n", i, C[i]);
    }

    // Cleanup
    clReleaseMemObject(d_A);
    clReleaseMemObject(d_B);
    clReleaseMemObject(d_C);
    clReleaseKernel(kernel);
    clReleaseProgram(program);
    clReleaseCommandQueue(queue);
    clReleaseContext(context);
    free(A);
    free(B);
    free(C);

    return 0;
}
```

OUTPUT

```
(base) PS C:\Users\Karunya\Documents\Sem 7 - LAs\GPA\Assignments> cmd /c '"C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools\Common7\Tools\VsDevCmd.bat" -arch=x64 && cl /I"C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.9\include" vector_addition.c "C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.9\lib\x64\OpenCL.lib" /Fe:vector_addition.exe'
**********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.14.13
** Copyright (c) 2025 Microsoft Corporation
**********************************************************************
Microsoft (R) C/C++ Optimizing Compiler Version 19.44.35215 for x64
Copyright (C) Microsoft Corporation.  All rights reserved.

vector_addition.c
cl_version.h: CL_TARGET_OPENCL_VERSION is not defined. Defaulting to 300 (OpenCL 3.0)
Microsoft (R) Incremental Linker Version 14.44.35215.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:vector_addition.exe
vector_addition.obj
"C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.9\lib\x64\OpenCL.lib"
(base) PS C:\Users\Karunya\Documents\Sem 7 - LAs\GPA\Assignments> .\vector_addition.exe
CPU Time: 42511.0000 ms
GPU/OpenCL Time: 25150.0000 ms
C[0] = 0.000000
C[1] = 2.000000
C[2] = 4.000000
C[3] = 6.000000
C[4] = 8.000000
C[5] = 10.000000
C[6] = 12.000000
C[7] = 14.000000
C[8] = 16.000000
C[9] = 18.000000
(base) PS C:\Users\Karunya\Documents\Sem 7 - LAs\GPA\Assignments>
```

Figure 1: Vector Addition Leveraging Heterogeneous Computing Using OpenCL.