گزارش بخش عملی دستورکار دوازدهم.

امیرحسین ادواری ۹۸۲٤۳۰۰٤ – زهرا حیدری ۹۸۲٤۳۰۲۰

شرح کد

```
7   // util functions for printing vectors
8   void print_vector(float *ptr, int size);
9   void print_vectors(float *zi1, float *zi2, float *zi3);
10
11  // Calculates y2-a, y2-b and 1/2(y4+(b-a)y2+ab) and stores them in zi1, zi2 and zo1
12  void f1(float a, float b, float *zi1, float *zi2, float *zo1);
13
14  // Calculates 1/2(zi1*zi2) and stores the result in zo2 (using assembly)
15  void div_asm(float *zi1, float *zi2, float *zo2);
16
17  // Copies the second array to first one :
18  void cp_fp_arrays(float *arr1, float *arr2, int size);
```

دو تابع کمکی برای پرینت کردن بردارها نوشته‌ایم که عملکرد آن‌ها واضح است.

تابع f1 برای محاسبه مقادیر خواسته شده در صورت سوال به زبان سی استفاده می‌شود.

تابع div_asm نیز مقدار خواسته شده در سوال ۲ را با دستورات اسمبلی (SSE) محاسبه می‌کند. (این دو تابع در ادامه شرح داده می‌شوند)

یک تابع کمکی دیگر برای کپی کردن یک آرایه در دیگری داریم که عملکرد آن واضح است.

```
94  void f1(float a, float b, float *zi1, float *zi2, float *zo1){
95
96      // fill y's
97      float y[50];
98      float step = 0.0;
99      for(int i = 0; i < 50; i++){
100         y[i] = step;
101         step += 0.2;
102     }
103
104     float y2;
105     for(int i = 0; i < 50; i++){
106         y2 = (y[i]*y[i]);
107         zi1[i] = y2 - a;
108         zi2[i] = y2 + b;
109         zo1[i] = 0.5 * (1/( zi1[i]*zi2[i]));
110     }
111 }
```

مشخصا در این تابع دو مقدار a و b و نیز سه وکتور zi1 و zi2 و zo1 را بعنوان ورودی گرفته و پس از انجام محاسبات آن‌هارا مقدار دهی می‌کنیم. بدین ترتیب که ابتدا وکتور y را با توجه به گام داده شده ایجاد می‌کنیم (هر عنصر از عنصر پیشین خود به اندازه گام تعیین شده بیشتر است)

سپس ۵۰ خانه ۳ وکتور ورودی را مقدار دهی میکنیم طبق صورت پروژه در zi1 مقادیر y2-a ، در zi2 مقادیر y2-b و در zo1 معکوس ۲برابر حاصل‌ضرت آن‌ها را قرار می‌دهیم.

```
56  void div_asm(float *zi1, float *zi2, float *zo2){
57      __declspec(align(16)) float zi1_aligned[50];
58      __declspec(align(16)) float zi2_aligned[50];
59      __declspec(align(16)) float zo2_aligned[52];
60      __declspec(align(16)) float ones[4] = { 1.0, 1.0, 1.0, 1.0};
61      __declspec(align(16)) float twos[4] = { 2.0, 2.0, 2.0, 2.0};
62
63      cp_fp_arrays(zi1_aligned, zi1, 50);
64      cp_fp_arrays(zi2_aligned, zi2, 50);
65
66      _asm {
67
68      mov ecx, 0
69  ITERATE:
70
71      movaps xmm0, oword ptr zi1_aligned[ecx]
72      movaps xmm1, oword ptr zi2_aligned[ecx]
73      movaps xmm2, oword ptr ones
74      movaps xmm3, oword ptr twos
75
76      mulps xmm0, xmm3 ; xmm0*2
77      mulps xmm0, xmm1 ; xmm0*xmm1
78      divps xmm2, xmm0 ; 1/xmm0
79
80      ;now xmm2 holds 1/(2*zi1*zi2)
81
82      movaps oword ptr zo2_aligned[ecx], xmm2
83
84      add ecx, 16
85      cmp ecx, 208
86
87      JE DONE
88      JMP short ITERATE
89  DONE:
90
91      }
92      cp_fp_arrays(zo2, zo2_aligned, 50);
93  }
```

در این تابع ابتدا طبق داکیومنت داده شده، آرایه‌های ورودی را در یک آرایه align شده می‌ریزیم. ۲وکتور ٤عنصری نیز برای ۱ و ۲ در نظر می‌گیریم (عبارت خواسته شده در سوال یک ضرب در ۲ و یک معکوس کردن نیاز دارد، از این دو وکتور برای این قسمت محاسبه استفاده م-کنیم)

بدین ترتیب، در هرمرحله ٤ عدد ممیز شناور از zi1 در xmm1 و ٤ عدد از zi2 در xmm2 میریزیم (میدانیم هر عدد ممیز شناور (single precision) ۳۲ بیت است از آنجا که xmm ها ۱۲۸ بیتی هستند، ٤ عدد ممیز شناور در آن‌ها می‌توان قرارداد)

سپس به طور چندتایی (با دستور mulps) هر ٤ عنصر xmm0 را دو برابر می‌کنیم (از طریق ضرب نظیر به نظیر با xmm3 که حاوی ٤ عنصر با مقدار ۲ می‌باشد) سپس عناصر آنرا نظیر به نظیر با xmm1 که ٤ عنصر از zi1 است ضرب می‌کنیم و در نتیجه را درخودش قرار می‌دهیم. در نهایت xmm2 را تقسیم چندتایی (نظیر به نظیر با دستور divps) با نتیجه بدست آمده می‌کنیم و حاصل را در ٤ خانه آرایه zo2 که حاوی نتیجه محاسبه است می‌ریزیم.

```
20  int main()
21  {
22
23      float a, b;
24      // Get Inputs
25      cprintf("Enter floating point numbers a, b\n");
26      cscanf("%f %f", &a, &b);
27      cprintf("%f %f\n", a, b);
28
29      // Define Vectors
30      float zi1[50];
31      float zi2[50];
32      float zo1[50];
33      float zo2[50];
34
35
36      f1(a, b, zi1, zi2, zo1);
37      cprintf("\nC program to calculate y2-a, y2-b and 1/2(y4+(b-a)y2+ab) \n\n");
38      print_vectors(zi1, zi2, zo1);
39
40      cprintf("\nassembly program to calculate 1/2(y4+(b-a)y2+ab) using y2-a, y2-b which has been calculated by c program: \n\n");
41      div_asm(zi1, zi2, zo2);
42      print_vectors(zi1, zi2, zo2);
43
44      cprintf("\n\n");
45      getch();
46      getch();
47      return 0;
48  }
```

در تابع main ورودی‌ها را گرفته، مقادیر y2-b و y2-a را با زبان c محاسبه کرده و در وکتورهای تعریف شده می‌ریزیم. معکوس ۲ برابر حاصل‌ضرب آن‌هارا نیز هم با زبان اسمبلی و هم با زبان c محاسبه می‌کنیم و نتایج را نشان می‌دهیم.