

Kevin Wittlinger "I pledge my honor that I have abided by the stevens honor system"

```
Register group: general
x0      0x0      0
x1      0x0      0
x2      0x410114  4260116
x3      0x41012c  4260140
x4      0x0      0
x5      0x0      0
x6      0x0      0
x7      0x0      0
x8      0x0      0
x9      0x0      0
x10     0x0      0
x11     0x0      0
x12     0x0      0
x13     0x0      0
x14     0x0      0
x15     0x0      0

Dot_product.s
B+ 8      MOV X0, 0 //Set register X0 to 0
9      MOV X1, 0 //Set register X1 to 0
10     ADR X2, vec1 //loads vec1 into register X2
11     ADR X3, vec2 //loads vec2 into register X3
B+> 12     ADR X4, dot //loads dot into register X7
13
14     LDR X5, [X2, 0] //Loads first element of vec1
15     LDR X6, [X3, 0] //Loads first element of vec2
16     MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
b+ 17     ADD X0, X0, X7 //add X0 and X7 to X0
18
19     ADD X2, X2, #8 // Move to next element of vec1
b+ 20     ADD X3, X3, #8 // Move to next element of vec2
21
22     LDR X5, [X2] //Loads second element of vec1
23     LDR X6, [X3] //Loads second element of vec2

native process 4433 In: _start
```

Breakpoint stopping once empty registers and data types are loaded into registers

```
Register group: general
x0      0x0      0
x1      0x0      0
x2      0x410114  4260116
x3      0x41012c  4260140
x4      0x410144  4260164
x5      0xa      10
x6      0x1      1
x7      0xa      10
x8      0x0      0
x9      0x0      0
x10     0x0      0
x11     0x0      0
x12     0x0      0
x13     0x0      0
x14     0x0      0
x15     0x0      0

Dot_product.s
5      global _start
6      _start:
7
B+ 8      MOV X0, 0 //Set register X0 to 0
9      MOV X1, 0 //Set register X1 to 0
10     ADR X2, vec1 //loads vec1 into register X2
11     ADR X3, vec2 //loads vec2 into register X3
B+> 12     ADR X4, dot //loads dot into register X7
13
14     LDR X5, [X2, 0] //Loads first element of vec1
15     LDR X6, [X3, 0] //Loads first element of vec2
16     MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
B+> 17     ADD X0, X0, X7 //add X0 and X7 to X0
18
19     ADD X2, X2, #8 // Move to next element of vec1
b+ 20     ADD X3, X3, #8 // Move to next element of vec2

native process 4433 In: _start
(gdb) c
Continuing.

Breakpoint 6, _start () at Dot_product.s:17
=> 0x0000000000000000 <_start+32>: 00 00 07 8b  add    x0, x0, x7
(gdb)
```

Now the code has loaded the first item within both vectors and multiplies these values then adds the result into X0

```

--Register group: general--
x0      0xa      10
x1      0x0      0
x2      0x41011c 4260124
x3      0x41012c 4260140
x4      0x410144 4260164
x5      0xa      10
x6      0x1      1
x7      0xa      10
x8      0x0      0
x9      0x0      0
x10     0x0      0
x11     0x0      0
x12     0x0      0
x13     0x0      0
x14     0x0      0
x15     0x0      0

Dot_product.s
14  LDR X5, [X2, 0] //Loads first element of vec1
15  LDR X6, [X3, 0] //Loads first element of vec2
16  MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
B+ 17  ADD X0, X0, X7 //add X0 and X7 to X0
18
19  ADD X2, X2, #8 // Move to next element of vec1
B+> 20  ADD X3, X3, #8 // Move to next element of vec2
21
22  LDR X5, [X2] //Loads second element of vec1
23  LDR X6, [X3] //Loads second element of vec2
24  MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
B+ 25  ADD X0, X0, X7 //add X0 and X7 to X0
26
27  ADD X2, X2, #8 // Move to next element of vec1
B+ 28  ADD X3, X3, #8 // Move to next element of vec2
29

native process 4433 In: _start L20 PC:
(gdb) c
Continuing.

Breakpoint 6, _start () at Dot_product.s:17
=> 0x000000004000d0 <_start+32>: 00 00 07 8b add x0, x0, x7
(gdb) c
Continuing.

Breakpoint 7, _start () at Dot_product.s:20
=> 0x000000004000d0 <_start+40>: 63 20 00 91 add x3, x3, #0x8
(gdb)

```

Moves both vector 1 and 2 to the next element

```

--Register group: general--
x0      0xa      10
x1      0x0      0
x2      0x41011c 4260124
x3      0x410134 4260148
x4      0x410144 4260164
x5      0x14     20
x6      0x2      2
x7      0x28     40
x8      0x0      0
x9      0x0      0
x10     0x0      0
x11     0x0      0
x12     0x0      0
x13     0x0      0
x14     0x0      0
x15     0x0      0

Dot_product.s
14  LDR X5, [X2, 0] //Loads first element of vec1
15  LDR X6, [X3, 0] //Loads first element of vec2
16  MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
B+ 17  ADD X0, X0, X7 //add X0 and X7 to X0
18
19  ADD X2, X2, #8 // Move to next element of vec1
B+ 20  ADD X3, X3, #8 // Move to next element of vec2
21
22  LDR X5, [X2] //Loads second element of vec1
23  LDR X6, [X3] //Loads second element of vec2
24  MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
B+> 25  ADD X0, X0, X7 //add X0 and X7 to X0
26
27  ADD X2, X2, #8 // Move to next element of vec1
B+ 28  ADD X3, X3, #8 // Move to next element of vec2
29

native process 4433 In: _start L25 PC:
(gdb) c
Continuing.

Breakpoint 6, _start () at Dot_product.s:17
=> 0x000000004000d0 <_start+32>: 00 00 07 8b add x0, x0, x7
(gdb) c
Continuing.

Breakpoint 7, _start () at Dot_product.s:20
=> 0x000000004000d0 <_start+40>: 63 20 00 91 add x3, x3, #0x8
(gdb) c
Continuing.

Breakpoint 8, _start () at Dot_product.s:25
=> 0x000000004000e8 <_start+56>: 00 00 07 8b add x0, x0, x7
(gdb)

```

loaded the second item within both vectors and multiplies these values then adds the result into X0

```

--Register group: general--
x0      0x32      50
x1      0x0       0
x2      0x410124  4260132
x3      0x410134  4260148
x4      0x410144  4260164
x5      0x14      20
x6      0x2       2
x7      0x28      40
x8      0x0       0
x9      0x0       0
x10     0x0       0
x11     0x0       0
x12     0x0       0
x13     0x0       0
x14     0x0       0
x15     0x0       0

Dot_product.s
21
22 LDR X5, [X2] //Loads second element of vec1
23 LDR X6, [X3] //Loads second element of vec2
24 MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
B+ 25
25 ADD X0, X0, X7 //add X0 and X7 to X0
26
27 ADD X2, X2, #8 // Move to next element of vec1
B+ 28
28 ADD X3, X3, #8 // Move to next element of vec2
29
30 LDR X5, [X2] //Loads third element of vec1
31 LDR X6, [X3] //Loads third element of vec2
32 MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
B+ 33
33 ADD X0, X0, X7 //add X0 and X7 to X0
34
35 STR X0, [X4] //Store the result into X4
36

Native process 4433 In: .start L28 PC: 0x
(gdb) c
Continuing.

Breakpoint 6, .start () at Dot_product.s:17
=> 0x000000004000d0: 00 00 07 8b add x0, x0, x7
(gdb) c
Continuing.

Breakpoint 7, .start () at Dot_product.s:20
=> 0x000000004000d8: 63 20 00 91 add x3, x3, #0x8
(gdb) c
Continuing.

Breakpoint 8, .start () at Dot_product.s:25
=> 0x000000004000e0: 00 00 07 8b add x0, x0, x7
(gdb) c
Continuing.

Breakpoint 9, .start () at Dot_product.s:28
=> 0x000000004000f0: 63 20 00 91 add x3, x3, #0x8
(gdb) c
Continuing.

```

Moves to the third element in both vectors

```

--Register group: general--
x0      0x32      50
x1      0x0       0
x2      0x410124  4260132
x3      0x41013c  4260156
x4      0x410144  4260164
x5      0x1e      30
x6      0x3       3
x7      0x5a      90
x8      0x0       0
x9      0x0       0
x10     0x0       0
x11     0x0       0
x12     0x0       0
x13     0x0       0
x14     0x0       0
x15     0x0       0

Dot_product.s
24 MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
B+ 25
25 ADD X0, X0, X7 //add X0 and X7 to X0
26
27 ADD X2, X2, #8 // Move to next element of vec1
B+ 28
28 ADD X3, X3, #8 // Move to next element of vec2
29
30 LDR X5, [X2] //Loads third element of vec1
31 LDR X6, [X3] //Loads third element of vec2
32 MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
B+ 33
33 ADD X0, X0, X7 //add X0 and X7 to X0
34
35 STR X0, [X4] //Store the result into X4
36
37 MOV X0, 0 // Set register X0 to 0 (return code)
38 MOV X0, 93 // Set register X0 to 93 (syscall number for exit)
B+ 39
39 SVC 0 // Invoke syscall to exit

Native process 4433 In: .start L33 PC: 0x
(gdb) c
Continuing.

Breakpoint 7, .start () at Dot_product.s:20
=> 0x000000004000d8: 63 20 00 91 add x3, x3, #0x8
(gdb) c
Continuing.

Breakpoint 8, .start () at Dot_product.s:25
=> 0x000000004000e0: 00 00 07 8b add x0, x0, x7
(gdb) c
Continuing.

Breakpoint 9, .start () at Dot_product.s:28
=> 0x000000004000f0: 63 20 00 91 add x3, x3, #0x8
(gdb) c
Continuing.

Breakpoint 10, .start () at Dot_product.s:33
=> 0x00000000400100: 00 00 07 8b add x0, x0, x7
(gdb) c
Continuing.

```

loaded the third item within both vectors and multiplies these values then adds the result into X0

```

--Register group: general
x0      0x0c      140
x1      0x0      0
x2      0x410124  4260132
x3      0x41013c  4260156
x4      0x410144  4260164
x5      0x1e      30
x6      0x3      3
x7      0x5a      90
x8      0x0      0
x9      0x0      0
x10     0x0      0
x11     0x0      0
x12     0x0      0
x13     0x0      0
x14     0x0      0
x15     0x0      0

--Dot product.s
24 MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
25 ADD X0, X0, X7 //add X0 and X7 to X0
26
27 ADD X2, X2, #8 // Move to next element of vec1
28 ADD X3, X3, #8 // Move to next element of vec2
29
30 LDR X5, [X2] //Loads third element of vec1
31 LDR X6, [X3] //Loads third element of vec2
32 MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
33 ADD X0, X0, X7 //add X0 and X7 to X0
34
35 STR X0, [X4] //Store the result into X4
36
37 MOV X0, 0 // Set register X0 to 0 (return code)
38 MOV X0, 93 // Set register X0 to 93 (syscall number for exit)
39 SVC 0 // Invoke syscall to exit

Native process 4433 In: _start L35 PC: 0x
(gdb) c
Continuing.

Breakpoint 8, _start () at Dot_product.s:25
=> 0x000000004000e0 <_start+56>: 00 00 07 8b add x0, x0, x7
(gdb) c
Continuing.

Breakpoint 9, _start () at Dot_product.s:28
=> 0x000000004000f0 <_start+64>: 63 20 00 91 add x3, x3, #0x8
(gdb) c
Continuing.

Breakpoint 10, _start () at Dot_product.s:33
=> 0x00000000400100 <_start+80>: 00 00 07 8b add x0, x0, x7
(gdb) c
Continuing.

Breakpoint 11, _start () at Dot_product.s:35
=> 0x00000000400104 <_start+84>: 80 00 00 f9 str x0, [x4]
(gdb) l

```

Stores the result from previous calculation into X0 register

```

(gdb) x/1dg &dot
0x410144: 140
(gdb)

```

Here is the result being printed

```

--Register group: general
x0      0x0      0
x1      0x0      0
x2      0x410124  4260132
x3      0x41013c  4260156
x4      0x410144  4260164
x5      0x1e      30
x6      0x3      3
x7      0x5a      90
x8      0x5a      93
x9      0x0      0
x10     0x0      0
x11     0x0      0
x12     0x0      0
x13     0x0      0
x14     0x0      0
x15     0x0      0

--Dot product.s
32 MUL X7, X5, X6 //Multiplies X5 and X6 stored in X7
33 ADD X0, X0, X7 //add X0 and X7 to X0
34
35 STR X0, [X4] //Store the result into X4
36
37 MOV X0, 0 // Set register X0 to 0 (return code)
38 MOV X0, 93 // Set register X0 to 93 (syscall number for exit)
39 SVC 0 // Invoke syscall to exit
40
41 .data
42 vec1: .quad 10, 20, 30
43 vec2: .quad 1, 2, 3
44 dot: .quad 0
45
46
47

Native process 4433 In: _start L39 PC: 0x400
(gdb) b 20
Breakpoint 13 at 0x400180: file Dot_product.s, line 37.
(gdb) c
Continuing.

Breakpoint 13, _start () at Dot_product.s:37
=> 0x00000000400180 <_start+80>: 00 00 00 d2 mov x0, #0x0 // 0
(gdb) x/2db $x10
0x0: Cannot access memory at address 0x0
(gdb) x/2db $x0c
Value can't be converted to integer.
(gdb) x/2db $x0e
Value can't be converted to integer.
(gdb) x/1dg &dot
0x410144: 140
(gdb) c
Continuing.

Breakpoint 12, _start () at Dot_product.s:39
=> 0x00000000400110 <_start+90>: 01 00 00 d4 svc #0x0
(gdb) l

```

The program then syscalls to exit the code