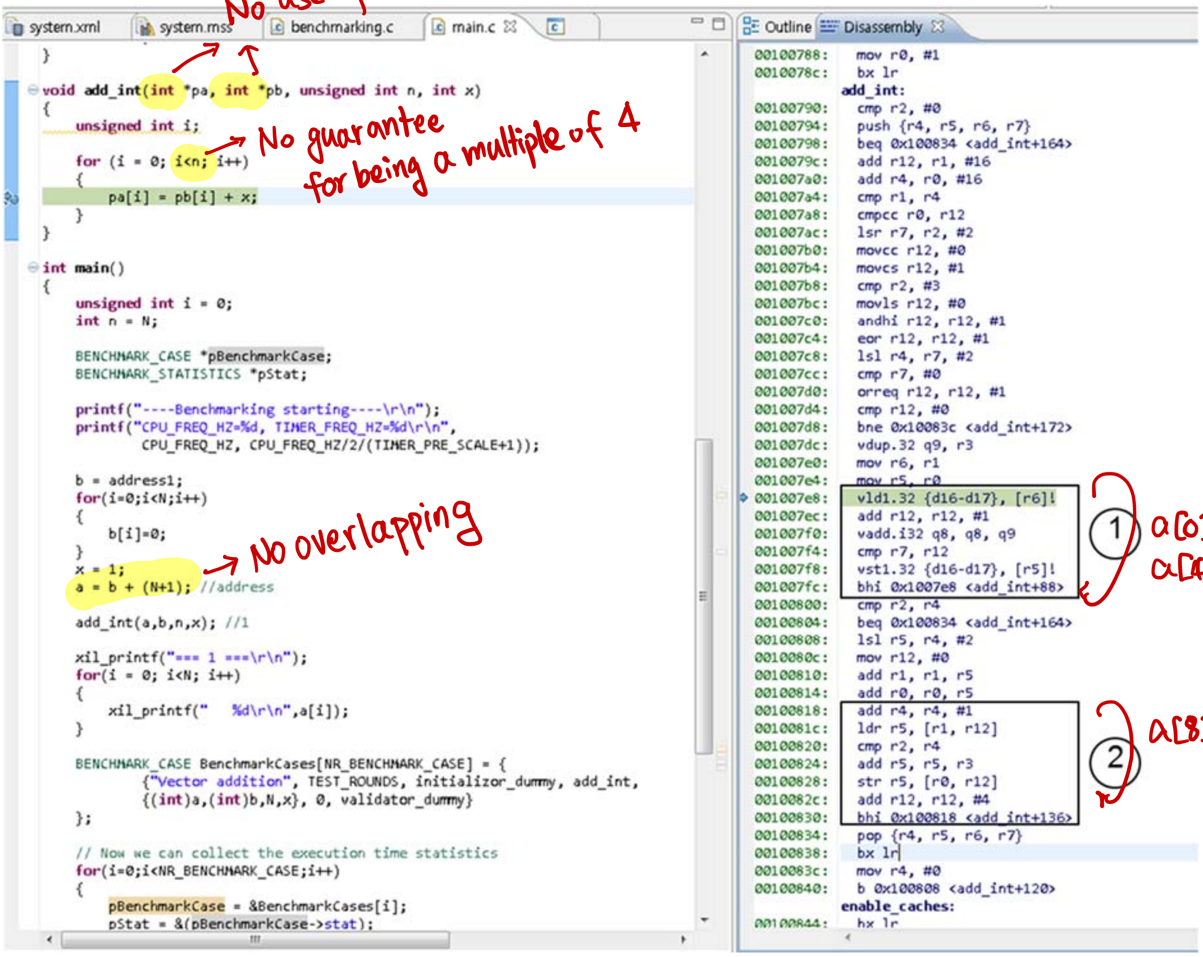
Student #: Name:

Quiz #4-1 2018 (5 min.)

Note 1: ***Return** the answer sheet (even the blank one) to show your attendance.*

Figure out how many times **each** of the two code segments (indicated by the two rectangles below) are executed in the entire program below (assuming that N is defined to be 9) [5pt].

No use of "restricted"



```
}

void add_int(int *pa, int *pb, unsigned int n, int x)
{
    unsigned int i;

    for (i = 0; i<n; i++)
    {
        pa[i] = pb[i] + x;
    }
}

int main()
{
    unsigned int i = 0;
    int n = N;

    BENCHMARK_CASE *pBenchmarkCase;
    BENCHMARK_STATISTICS *pStat;

    printf("----Benchmarking starting----\r\n");
    printf("CPU_FREQ_HZ=%d, TIMER_FREQ_HZ=%d\r\n",
            CPU_FREQ_HZ, CPU_FREQ_HZ/2/(TIMER_PRE_SCALE+1));

    b = address1;
    for(i=0;i<N;i++)
    {
        b[i]=0;
    }
    x = 1;
    a = b + (N+1); //address

    add_int(a,b,n,x); //1

    xil_printf("=== 1 ===\r\n");
    for(i = 0; i<N; i++)
    {
        xil_printf("   %d\r\n",a[i]);
    }

    BENCHMARK_CASE BenchmarkCases[NR_BENCHMARK_CASE] = {
            {"Vector addition", TEST_ROUNDS, initializor_dummy, add_int,
            {(int)a,(int)b,N,x}, 0, validator_dummy}
    };

    // Now we can collect the execution time statistics
    for(i=0;i<NR_BENCHMARK_CASE;i++)
    {
        pBenchmarkCase = &BenchmarkCases[i];
        pStat = &(pBenchmarkCase->stat);
```

No guarantee for being a multiple of 4

No overlapping

```
00100788:   mov r0, #1
0010078c:   bx lr
          add_int:
00100790:   cmp r2, #0
00100794:   push {r4, r5, r6, r7}
00100798:   beq 0x100834 <add_int+164>
0010079c:   add r12, r1, #16
001007a0:   add r4, r0, #16
001007a4:   cmp r1, r4
001007a8:   cmpcc r0, r12
001007ac:   lsr r7, r2, #2
001007b0:   movcc r12, #0
001007b4:   movcs r12, #1
001007b8:   cmp r2, #3
001007bc:   movls r12, #0
001007c0:   andhi r12, r12, #1
001007c4:   eor r12, r12, #1
001007c8:   lsl r4, r7, #2
001007cc:   cmp r7, #0
001007d0:   orreq r12, r12, #1
001007d4:   cmp r12, #0
001007d8:   bne 0x10083c <add_int+172>
001007dc:   vdup.32 q9, r3
001007e0:   mov r6, r1
001007e4:   mov r5, r0
001007e8:   vld1.32 {d16-d17}, [r6]!
001007ec:   add r12, r12, #1
001007f0:   vadd.i32 q8, q8, q9
001007f4:   cmp r7, r12
001007f8:   vst1.32 {d16-d17}, [r5]!
001007fc:   bhi 0x1007e8 <add_int+88>
00100800:   cmp r2, r4
00100804:   beq 0x100834 <add_int+164>
00100808:   lsl r5, r4, #2
0010080c:   mov r12, #0
00100810:   add r1, r1, r5
00100814:   add r0, r0, r5
00100818:   add r4, r4, #1
0010081c:   ldr r5, [r1, r12]
00100820:   cmp r2, r4
00100824:   add r5, r5, r3
00100828:   str r5, [r0, r12]
0010082c:   add r12, r12, #4
00100830:   bhi 0x100818 <add_int+136>
00100834:   pop {r4, r5, r6, r7}
00100838:   bx lr
0010083c:   mov r4, #0
00100840:   b 0x100808 <add_int+120>
          enable_caches:
00100844:   bx lr
```

a[0]~a[3]
a[4]~a[7]

a[8]

Code segment #1: (2)

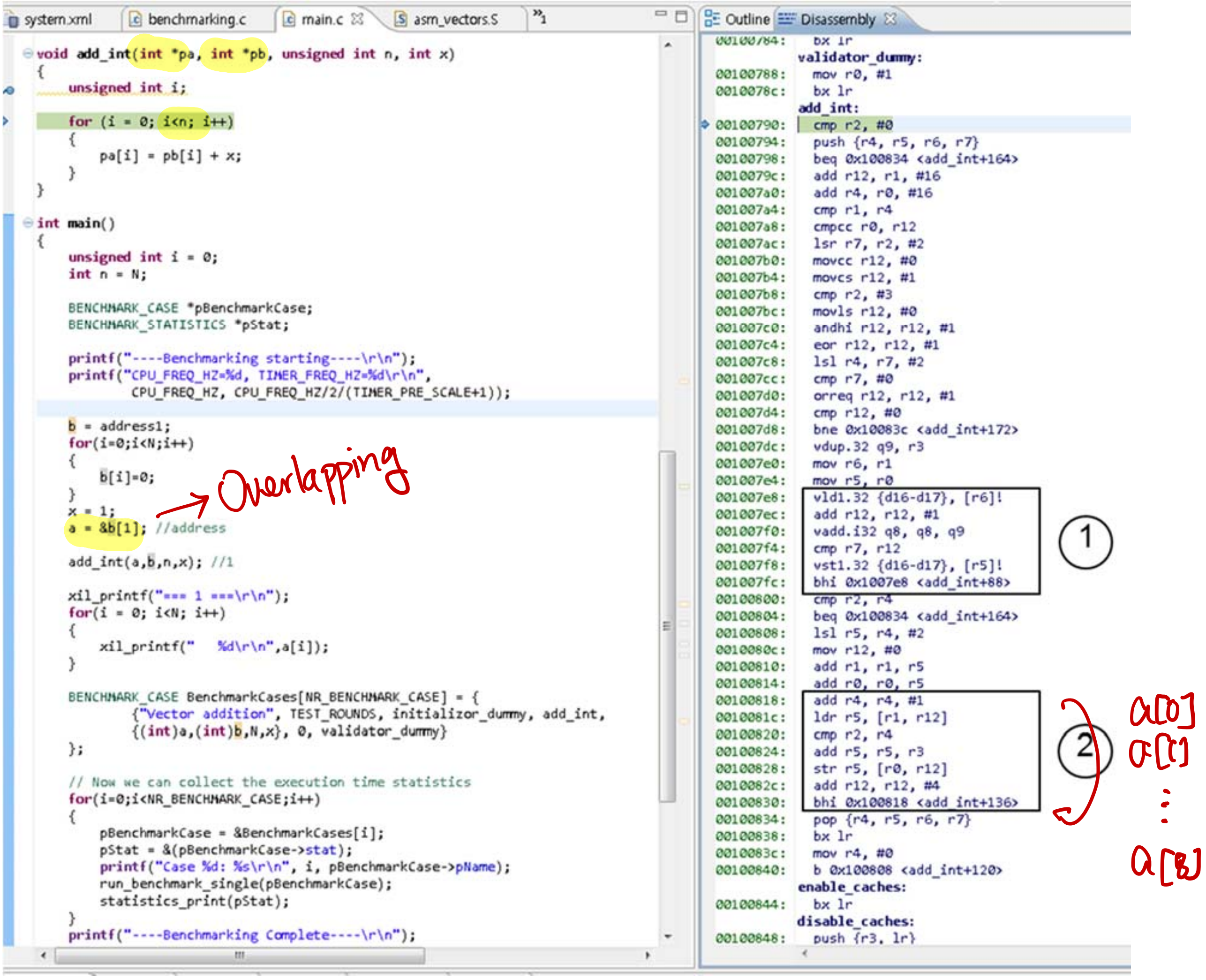
Code segment #2: (1)

Student #: Name:

Quiz #4-2 2018 (5 min.)

Note 1: ***Return** the answer sheet (even the blank one) to show your attendance.*

Figure out how many times **each** of the two code segments (indicated by the two rectangles below) are executed in the entire program below (assuming that `N` is defined to be 9) [5pt].



```c
void add_int(int *pa, int *pb, unsigned int n, int x)
{
    unsigned int i;

    for (i = 0; i<n; i++)
    {
        pa[i] = pb[i] + x;
    }
}

int main()
{
    unsigned int i = 0;
    int n = N;

    BENCHMARK_CASE *pBenchmarkCase;
    BENCHMARK_STATISTICS *pStat;

    printf("----Benchmarking starting----\r\n");
    printf("CPU_FREQ_HZ=%d, TIMER_FREQ_HZ=%d\r\n",
            CPU_FREQ_HZ, CPU_FREQ_HZ/2/(TIMER_PRE_SCALE+1));

    b = address1;
    for(i=0;i<N;i++)
    {
        b[i]=0;
    }
    x = 1;
    a = &b[1]; //address

    add_int(a,b,n,x); //1

    xil_printf("=== 1 ===\r\n");
    for(i = 0; i<N; i++)
    {
        xil_printf("   %d\r\n",a[i]);
    }

    BENCHMARK_CASE BenchmarkCases[NR_BENCHMARK_CASE] = {
            {"Vector addition", TEST_ROUNDS, initializor_dummy, add_int,
            {(int)a,(int)b,N,x}, 0, validator_dummy}
    };

    // Now we can collect the execution time statistics
    for(i=0;i<NR_BENCHMARK_CASE;i++)
    {
        pBenchmarkCase = &BenchmarkCases[i];
        pStat = &(pBenchmarkCase->stat);
        printf("Case %d: %s\r\n", i, pBenchmarkCase->pName);
        run_benchmark_single(pBenchmarkCase);
        statistics_print(pStat);
    }
    printf("----Benchmarking Complete----\r\n");
```

```
00100784:  bx lr
          validator_dummy:
00100788:  mov r0, #1
0010078c:  bx lr
          add_int:
00100790:  cmp r2, #0
00100794:  push {r4, r5, r6, r7}
00100798:  beq 0x100834 <add_int+164>
0010079c:  add r12, r1, #16
001007a0:  add r4, r0, #16
001007a4:  cmp r1, r4
001007a8:  cmpcc r0, r12
001007ac:  lsr r7, r2, #2
001007b0:  movcc r12, #0
001007b4:  movcs r12, #1
001007b8:  cmp r2, #3
001007bc:  movls r12, #0
001007c0:  andhi r12, r12, #1
001007c4:  eor r12, r12, #1
001007c8:  lsl r4, r7, #2
001007cc:  cmp r7, #0
001007d0:  orreq r12, r12, #1
001007d4:  cmp r12, #0
001007d8:  bne 0x10083c <add_int+172>
001007dc:  vdup.32 q9, r3
001007e0:  mov r6, r1
001007e4:  mov r5, r0
001007e8:  vld1.32 {d16-d17}, [r6]!
001007ec:  add r12, r12, #1
001007f0:  vadd.i32 q8, q8, q9
001007f4:  cmp r7, r12
001007f8:  vst1.32 {d16-d17}, [r5]!
001007fc:  bhi 0x1007e8 <add_int+88>
00100800:  cmp r2, r4
00100804:  beq 0x100834 <add_int+164>
00100808:  lsl r5, r4, #2
0010080c:  mov r12, #0
00100810:  add r1, r1, r5
00100814:  add r0, r0, r5
00100818:  add r4, r4, #1
0010081c:  ldr r5, [r1, r12]
00100820:  cmp r2, r4
00100824:  add r5, r5, r3
00100828:  str r5, [r0, r12]
0010082c:  add r12, r12, #4
00100830:  bhi 0x100818 <add_int+136>
00100834:  pop {r4, r5, r6, r7}
00100838:  bx lr
0010083c:  mov r4, #0
00100840:  b 0x100808 <add_int+120>
          enable_caches:
00100844:  bx lr
          disable_caches:
00100848:  push {r3, lr}
```

Code segment #1: (0)

Code segment #2: (9)