Computer Organisation
Lab - 4

Team Members

Ishwar Govind 111901024
Abraham Mathew Illickan 111901003

Testing on

even-odd.asm

```
1.   .data
2. a:
3.   11
4.   .text
5. main:
6.   load %x0, $a, %x4
7.   andi %x4, 1, %x5
8.   muli %x5, 2, %x6
9.   subi %x6, 1, %x10
10.  end
```

even-odd.stats

```
1. Number of instructions executed = 5
2. Number of cycles taken = 25
```

palindrome.asm

```
1.   .data
2. a:
3.          12321
4.   .text
5. main:
6.          load %x0, $a, %x3
7.          addi %x3, 0, %x4
8.          addi %x0, 0, %x5
9. loop:
10.         beq %x4, 0, check
11.         muli %x5, 10, %x5
12.         divi %x4, 10, %x6
13.         muli %x6, 10, %x6
14.         sub %x4, %x6, %x7
15.         add %x5, %x7, %x5
16.         divi %x4, 10, %x4
```

```
17.          jmp loop
18.  check:
19.          beq %x3, %x5, sus
20.  fail:
21.          subi %x0, 1, %x10
22.          end
23.  sus:
24.          addi %x0, 1, %x10
25.          end
```

palindrome.stats

```
1. Number of instructions executed = 47
2. Number of cycles taken = 235
```

prime.asmr

```
1.   .data
2. a:
3.           6469
4.   .text
5. main:
6.           load %x0, $a, %x3
7.           addi %x0, 1, %x5
8.           beq %x3, %x5, notprime
9. loop:
10.          addi %x5, 1, %x5
11.          mul %x5, %x5, %x8
12.          bgt %x8, %x3, prime
13.  modcheck:
14.          div %x3, %x5, %x6
15.          mul %x5, %x6, %x6
16.          sub %x3, %x6, %x6
17.          beq %x6, %x0, notprime
18.          jmp loop
19.  prime:
20.          addi %x0, 1, %x10
21.          end
22.  notprime:
23.          subi %x0, 1, %x10
24.          end
```

prime.stats

```
1. Number of instructions executed = 640
2. Number of cycles taken = 3200
```

fibonacci.asm

```
1.   .data
2. n:
3.   10
4.   .text
5. main:
```

```
6.          load %x0, $n, %x4
7.          addi %x0, 0, %x5
8.          addi %x0, 1, %x6
9.          addi %x0, 0, %x8
10.         addi %x0, 65535, %x9
11. loop:
12.         beq %x8, %x4, endl
13.         store %x5, 0, %x9
14.         add %x5, %x6, %x7
15.         add %x0, %x6, %x5
16.         add %x0, %x7, %x6
17.         addi %x8, 1, %x8
18.         subi %x9, 1, %x9
19.         jmp loop
20. endl:
21.         end
```

fibonacci.stats

```
1. Number of instructions executed = 87
2. Number of cycles taken = 435
```

descending.asm

```
1.   .data
2. a:
3.          70
4.          80
5.          40
6.          20
7.          10
8.          30
9.          50
10.         60
11. n:
12.         8
13.         .text
14. main:
15.         load %x0, $n, %x3
16.         add %x0, %x0, %x4
17. loopi:
18.         bgt %x4, %x3, endi
19.         add %x0, %x4, %x5
20.         add %x0, %x4, %x6
21.         load %x6, $a, %x7
22. loopj:
23.         bgt %x5, %x3, endj
24.         load %5, $a, %x8
25.         bgt %x8, %x7, update
26. updated:
27.         addi %x5, 1, %x5
28.         jmp loopj
29. endj:
30.         load %x4, $a, %x9
31.         store %x7, $a, %x4
32.         store %x9, $a, %x6
33.         addi %x4, 1, %x4
34.         jmp loopi
```

```
35.  update:
36.          add %x8, %x0, %x7
37.          add %x5, %x0, %x6
38.          jmp updated
39.  endi:
40.          end
```

descending.stats

```
1. Number of instructions executed = 340
2. Number of cycles taken = 1700
```