Benjamin Smith
Email: bxs566@case.edu
Course: CSDS 337 - Compiler Design
Instructor: Dr. Vipin Chaudhary

Problem Set - 6
ID: 3559750
Term: Spring 2024
Due Date: 24^{th} April, 2024

Number of hours delay for this Problem Set: 0.
Cumulative number of hours delay so far: 56.

I discussed this homework with: No one.

Problem 1 - 10 points
Generate code for the following three-address statements assuming `a` and `b` are arrays whose elements are 4-byte values. The four-statement sequence

```
x = a[i]
y = b[j]
a[i] = y
b[j] = x
```

Solution:

```
LD R1, i
MUL R1, R1, 4
LD R2, a(R1)
LD R3, j
MUL R3, R3, 4
LD R4, b(R3)
ST a(R1), R4
ST b(R3), R2
```

Problem 2 - 10 points
Determine the cost of executing the following.

1.
```
LD R0, c
LD R1, i
MUL R1, R1, 8
ST a(R1), R0
```

2.
```
LD R0, p
LD R1, 0(R0)
ST x, R1
```

Solution:

1. $2 + 2 + 2 + 2 = 8$
2. $2 + 2 + 2 = 6$

Problem 3 - 10 points
Below is code to count the number of primes from 2 to n , using the sieve method on a suitably large array a. That is, a [i] is TRUE at the end only if there is no prime $\sqrt{i}$ or less that evenly divides i. We initialize all a[i] to TRUE and then set a[j] to FALSE if we find a divisor of j.

```
for (i=2; i<=n; i++)
    a[i] = TRUE;
count = 0;
s = sqrt(n);
for (i=2; i<=s; i++)
    if (a[i]) /* i has been found to be a prime */ {}
        count++;
        for (j=2*i; j<=n; j = j+i)
            a[j] = FALSE; /* no multiple of i is a prime */
    }
```

a Translate the program into three-address statements of the type we have been using in this section. Assume integers require 4 bytes.

b Construct the flow graph for your code from (a)

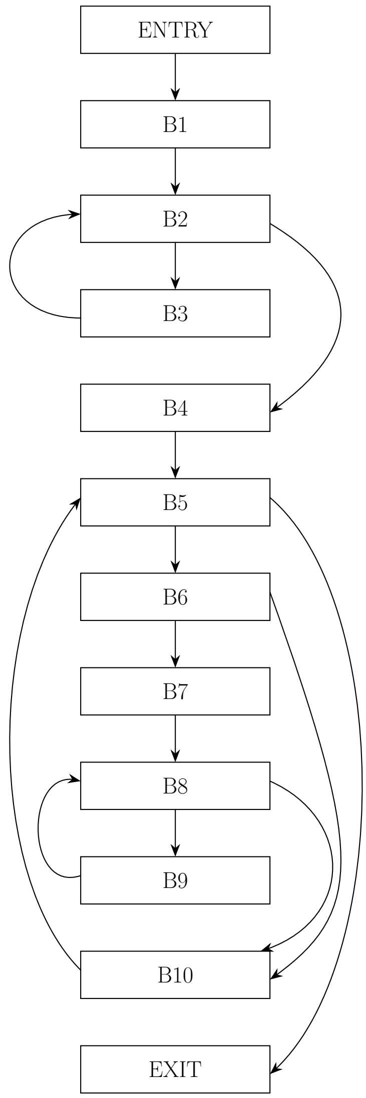
c Identify the loops in your flow graph from (b).

Solution:

a

```
B1      1)  i = 2

B2      2)  if i > n goto(7)

B3      3)  t1 = i * 4
        4)  a[t1] = TRUE
        5)  i = i + 1
        6)  goto(2)

B4      7)  count = 0
        8)  s = sqrt(n)
        9)  i = 2

B5      10)  if i > s goto(22)

B6      11)  t2 = i * 4
        12)  ifFalse a[t2] goto(20)

B7      13)  count = count + 1
        14)  j = 2 * i

B8      15)  if j > n goto(20)

B9      16)  t3 = j * 4
        17)  a[t3] = FALSE
        18) j = j + i
        19)  goto(15)

B10     20)  i = i + 1
        21)  goto(10)
```

b Graph:



c B2, B3
B5, B6, B10
B5, B6, B7, B8, B10
B8, B9

Problem 4 - 10 points
Suppose a basic block is formed from the C assignment statements

```
x = a + b + c + d + e + f;
y = a + c + e;
```

a Give the three-address statements (only one addition per statement) for this block.

b Use the associative and commutative laws to modify the block to use the fewest possible number of instructions, assuming both x and y are live on exit from the block.

Solution:

a

```
t1 = a + b
t2 = t1 + c
t3 = t2 + d
t4 = t3 + e
t5 = t4 + f
x = t5
t6 = a + c
t7 = c + e
y = t6 + t7
```

b

```
t1 = a + c
t2 = t1 + e
y = t2
t3 = t2 + b
t4 = t3 + d
t5 = t4 + f
x = t5
```

Problem 5 - 20 points
Consider the expression $(a - b) + e * (c + d)$.

a Generate optimized code using three registers.

b Generate optimized code using two registers.

Solution:

a

```
LD  R3, d
LD  R2, c
ADD R3, R2, R3
LD  R2, e
MUL R3, R2, R3
LD  R2, b
LD  R1, a
SUB R2, R1, R2
ADD R3, R2, R3
```

b

```
LD  R2, d
LD  R1, c
ADD R2, R1, R2
LD  R1, e
MUL R2, R1, R2
ST  t3, R2
LD  R2, b
LD  R1, a
SUB R2, R1, R2
LD  R1, t3
ADD R2, R2, R1
```
