COL226 Assignment 1
Basic Decimal Integer Machine

Rishabh Dhiman

13 January 2022

1 Absolute Value

1. ```
   cin >> x;
   if (x <= 0) x = -x;
   cout << x;
   ```

2. Get rid of constants and unary operators.

   ```
   zero = 0;
   cin >> x;
   if (x <= zero) x = zero - x;
   cout << x;
   ```

3. Write branches and loops as goto

   ```
   zero = 0;
   cin >> x;
   if (x > zero) goto OUT;
   x = zero - x;
   OUT: cout << x;
   ```

4. Write as code/mem state machine.

```
0  mem[0] := 0;
1  input: mem[1];
2  mem[2] := mem[1] > mem[0];
3  if mem[2] goto code[5];
4  mem[1] := mem[0] - mem[1];
5  output: mem[1];
6  halt;
```

5. Convert to BDIM with wildcards.
```

```
0  (16,0,_,0)
1  (1,_,_,1)
2  (12,1,0,2)
3  (13,2,_,5)
4  (7,0,1,1)
5  (15,1,_,_)
6  (0,_,_,_)
```

2 Arithmetic Progression

$$\sum_{i=0}^{n-1}(a+id) = na + \frac{n(n-1)}{2}d$$

1. ```
   cin >> a >> d >> n;
   cout << (n * a + n * (n - 1) / 2 * d);
   ```

2. Get rid of temporary values.

   ```
   cin >> a;
   cin >> d;
   cin >> n;
   x = 1;
   y = n - x;
   y = n * y;
   y = y * d;
   x = 2;
   y = y / x;
   x = n * a;
   x = x + y;
   cout << x;
   ```

3. Write as code/mem state machine.

```
0  input: mem[0];
1  input: mem[1];
2  input: mem[2];
3  mem[3] := 1;
4  mem[4] := mem[2] - mem[3];
5  mem[4] := mem[2] * mem[4];
6  mem[4] := mem[4] * mem[1];
7  mem[3] := 2;
8  mem[4] := mem[4] / mem[3];
9  mem[3] := mem[2] * mem[0];
10 mem[3] := mem[3] + mem[4];
```
```

```
11  output: mem[3];
12  halt;
```

4. Convert to assembly-BDIM

```
0   (input,_,_,0)
1   (input,_,_,1)
2   (input,_,_,2)
3   (read,1,_,3)
4   (sub,2,3,4)
5   (mul,2,4,4)
6   (mul,4,1,4)
7   (read,2,_,3)
8   (div,4,3,4)
9   (mul,2,0,3)
10  (add,3,4,3)
11  (output,3,_,_)
12  (halt,_,_,_)
```

5. Convert BDIM with strings to BDIM using code.

```
0   (1,37,37,0)
1   (1,37,37,1)
2   (1,37,37,2)
3   (16,1,37,3)
4   (7,2,3,4)
5   (8,2,4,4)
6   (8,4,1,4)
7   (16,2,37,3)
8   (9,4,3,4)
9   (8,2,0,3)
10  (6,3,4,3)
11  (15,3,37,37)
12  (0,37,37,37)
```

3 Factorial

1.
```
cin >> n;
f = 1;
i = 0;
while (i < n) {
        ++i;
        f = f * i;
}
```

2. Write branches and loops as goto

```
cin >> n;
f = 1;
i = 0;
L: i = i + 1;
f = f * i;
if (i < n) goto L;
cout << f;
```

3. Get rid of temporary values and constants.

```
cin >> n;
one = 1;
f = 1;
i = 0;
L: i = i + one;
f = f * i;
b = n > i;
if b goto L;
cout << f;
```

4. Write as code/mem state machine.

```
0  input: mem[0];
1  mem[1] := 1;
2  mem[2] := 1;
3  mem[3] := 0;
4  mem[3] := mem[3] + mem[1];
5  mem[2] := mem[2] * mem[3];
6  mem[4] := mem[0] > mem[3];
7  if mem[4] goto code[4];
8  output: mem[2];
9  halt;
```

5. Convert to BDIM-string

```
0  (input,_,_,0)
1  (read,1,_,1)
2  (read,1,_,2)
3  (read,0,_,3)
4  (add,3,1,3)
5  (mul,2,3,2)
6  (gt,0,3,1)
7  (if,4,_,4)
8  (output,2,_,_)
9  (halt,_,_,_)
```

6. Convert BDIM-string to BDIM using code.

```
0  (1,37,37,0)
1  (16,1,37,1)
2  (16,1,37,2)
3  (16,0,37,3)
4  (6,3,1,3)
5  (8,2,3,2)
6  (12,0,3,1)
7  (13,1,37,4)
8  (15,2,37,37)
9  (0,37,37,37)
```

4 Fibonacci

1. ```
   cin >> n;
   a = 0;
   b = 1;
   for (int i = 0; i < n; ++i) {
           a = a + b;
           c = a;
           a = b;
           b = c;
   }
   cout << a;
   ```

2. Write branches and loops as goto

   ```
   cin >> n;
   a = 0;
   b = 1;
   i = 1;
   L: if (i > n) goto E;
   a = a + b;
   c = a;
   a = b;
   b = c;
   i = i + 1;
   goto L;
   E: cout << a;
   ```

3. Get rid of temporary values and constants.

   ```
   cin >> num;
   a = 0;
   b = 1;
   i = 1;
   ```
   ```

```
one = 1;
L: cmp = i > num;
if (cmp) goto E;
a = a + b;
c = a;
a = b;
b = c;
i = i + one;
goto L;
E: cout << a;
```

4. Write as code/mem state machine.

```
0  input: mem[0];
1  mem[1] := 0;
2  mem[2] := 1;
3  mem[3] := 1;
4  mem[4] := 1;
5  mem[5] := mem[3] > mem[0];
6  if (mem[5]) goto code[13];
7  mem[1] := mem[1] + mem[2];
8  mem[6] := mem[1];
9  mem[1] := mem[2];
10 mem[2] := mem[6];
11 mem[3] := mem[3] + mem[4];
12 goto code[5];
13 output: mem[1];
14 halt;
```

5. Convert to assembly-BDIM

```
0  (input,_,_,0)
1  (read,0,_,1)
2  (read,1,_,2)
3  (read,1,_,3)
4  (read,1,_,4)
5  (gt,3,0,5)
6  (if,5,_,13)
7  (add,1,2,1)
8  (move,1,_,6)
9  (move,2,_,1)
10 (move,6,_,2)
11 (add,3,4,3)
12 (goto,_,_,5)
13 (output,1,_,_)
14 (halt,_,_,_)
```

6. Convert BDIM with strings to BDIM using code.

```
0   (1,37,37,0)
1   (16,0,37,1)
2   (16,1,37,2)
3   (16,1,37,3)
4   (16,1,37,4)
5   (12,3,0,5)
6   (13,5,37,13)
7   (6,1,2,1)
8   (2,1,37,6)
9   (2,2,37,1)
10  (2,6,37,2)
11  (6,3,4,3)
12  (14,37,37,5)
13  (15,1,37,37)
14  (0,37,37,37)
```

5 GCD

1. ```
   cin >> x >> y;
   if (x <= 0) x = -x;
   if (y <= 0) y = -y;
   while (y != 0) {
           x = x % y;
           z = x;
           x = y;
           y = z;
   }
   cout << x;
   ```

2. Write branches and loops as goto

   ```
   cin >> x >> y;
   if (x <= 0) x = -x;
   if (y <= 0) y = -y;
   L: if (y == 0) goto E;
   x = x % y;
   z = x;
   x = y;
   y = z;
   goto L;
   E: cout << x;
   ```

3. Get rid of temporary values and constants.
   ```

```
cin >> x;
cin >> y;
zero = 0;
cmp = x > zero;
if (cmp) goto A;
x = zero - x;
A: cmp = y > zero;
if (cmp) goto L;
y = zero - y;
L: cmp = y == zero;
if (cmp) goto E;
x = x % y;
z = x;
x = y;
y = z;
goto L;
E: cout << x;
```

4. Write as code/mem state machine.

```
0  input: mem[1];
1  input: mem[2];
2  mem[0] := 0;
3  mem[4] := mem[1] > mem[0];
4  if (mem[4]) goto code[6];
5  mem[1] := mem[0] - mem[1];
6  mem[4] := mem[2] > mem[0];
7  if (mem[4]) goto code[9];
8  mem[2] := mem[0] - mem[2];
9  mem[4] := mem[2] = mem[0];
10 if (mem[4]) goto code[16];
11 mem[1] := mem[1] mod mem[2];
12 mem[3] := mem[1];
13 mem[1] := mem[2];
14 mem[2] := mem[3];
15 goto code[9];
16 output: mem[1];
17 halt;
```

5. Convert to assembly-BDIM

```
0  (input,_,_,1)
1  (input,_,_,2)
2  (read,0,_,0)
3  (gt,1,0,4)
4  (if,4,_,6)
```

```
 5  (sub,0,1,1)
 6  (gt,2,0,4)
 7  (if,4,_,9)
 8  (sub,0,2,2)
 9  (eq,2,0,4)
10  (if,4,_,16)
11  (mod,1,2,1)
12  (move,1,_,3)
13  (move,2,_,1)
14  (move,3,_,2)
15  (goto,_,_,9)
16  (output,1,_,_)
17  (halt,_,_,_)
```

6. Convert BDIM with strings to BDIM using code.

```
 0  (1,37,37,1)
 1  (1,37,37,2)
 2  (16,0,37,0)
 3  (12,1,0,4)
 4  (13,4,37,6)
 5  (7,0,1,1)
 6  (12,2,0,4)
 7  (13,4,37,9)
 8  (7,0,2,2)
 9  (11,2,0,4)
10  (13,4,37,16)
11  (10,1,2,1)
12  (2,1,37,3)
13  (2,2,37,1)
14  (2,3,37,2)
15  (14,37,37,9)
16  (15,1,37,37)
17  (0,37,37,37)
```

6 Reverse

1. ```
   cin >> x;
   y = 0;
   while (x != 0) {
           y = 10 * y + (x % 10);
           x /= 10;
   }
   cout << y;
   ```
   ```

2. Write branches and loops as goto

```
cin >> x;
y = 0;
L: if (x == 0) goto E;
y = 10 * y + (x % 10);
x /= 10;
goto L;
E: cout << y;
```

3. Get rid of temporary values and constants.

```
zero = 0;
ten = 10;
cin >> x;
y = 0;
L: cmp = x == zero;
if (cmp) goto E;
y = ten * y;
mod = x % ten;
y = y + mod;
x = x / ten;
goto L;
E: cout << y;
```

4. Write as code/mem state machine.

```
0  mem[0] := 0;
1  mem[1] := 10;
2  input: mem[2];
3  mem[3] := 0;
4  mem[4] := mem[2] = mem[0];
5  if (mem[4]) goto code[11];
6  mem[3] := mem[1] * mem[3];
7  mem[4] := mem[2] % mem[1];
8  mem[3] := mem[3] + mem[4];
9  mem[2] := mem[2] / mem[1];
10 goto code[4];
11 output: mem[3];
```

5. Convert to assembly-BDIM

```
0  (read,0,_,0)
1  (read,10,_,1)
2  (input,_,_,2)
3  (read,0,_,3)
4  (eq,2,0,4)
```

```
5  (if,4,_,11)
6  (mul,1,3,3)
7  (mod,2,1,4)
8  (add,3,4,3)
9  (div,2,1,2)
10 (goto,_,_,4)
11 (output,3,_,_)
12 (halt,_,_,_)
```

6. Convert BDIM with strings to BDIM using code.

```
0  (16,0,37,0)
1  (16,10,37,1)
2  (1,37,37,2)
3  (16,0,37,3)
4  (11,2,0,4)
5  (13,4,37,11)
6  (8,1,3,3)
7  (10,2,1,4)
8  (6,3,4,3)
9  (9,2,1,2)
10 (14,37,37,4)
11 (15,3,37,37)
12 (0,37,37,37)
```

7 Russian Multiplication

1. ```
   cin >> x >> y;
   res = 0;
   while (y != 0) {
           if (y % 2 == 1)
                   res += x;
           x = x + x;
           y /= 2;
   }
   cout << res;
   ```

2. Write branches and loops as goto

   ```
   cin >> x >> y;
   res = 0;
   if (y > 0) goto L;
   x = -x;
   L: if (y == 0) goto E;
      if (y % 2 == 0) goto SKIP;
   ```
   ```

```
   res += x;
SKIP: x = x + x;
   y = y / 2;
   goto L;
E: cout << res;
```

3. Get rid of temporary values and constants.

```
zero = 0;
two = 2;
cin >> x;
cin >> y;
res = 0;
cmp = y > zero;
if (cmp) goto L;
x = zero - x;
y = zero - y;
L: cmp = y == zero;
if (cmp) goto E;
cmp = y % two;
cmp = cmp == zero;
if (cmp) goto SKIP;
res = res + x;
SKIP: x = x + x;
y = y / two;
goto L;
E: cout << res;
```

4. Write as code/mem state machine.

```
0  mem[0] := 0;
1  mem[1] := 2;
2  input: mem[2];
3  input: mem[3];
4  mem[4] := 0;
5  mem[5] := mem[3] > mem[0];
6  if (mem[5]) goto code[9];
7  mem[2] := mem[0] - mem[2];
8  mem[3] := mem[0] - mem[3];
9  L: mem[5] := mem[3] = mem[0];
10 if (mem[5]) goto code[18];
11 mem[5] := mem[3] % mem[1];
12 mem[5] := mem[5] = mem[0];
13 if (mem[5]) goto code[15];
14 mem[4] := mem[4] + mem[2];
15 SKIP: mem[2] := mem[2] + mem[2];
```

```
16  mem[3] := mem[3] / mem[1];
17  goto code[9];
18  E: output: mem[4];
19  halt;
```

5. Convert to assembly-BDIM

```
 0  (read,0,_,0)
 1  (read,2,_,1)
 2  (input,_,_,2)
 3  (input,_,_,3)
 4  (read,0,_,4)
 5  (gt,3,0,5)
 6  (if,5,_,9)
 7  (sub,0,2,2)
 8  (sub,0,3,3)
 9  (eq,3,0,5)
10  (if,5,_,18)
11  (mod,3,1,5)
12  (eq,5,0,5)
13  (if,5,_,15)
14  (add,4,2,4)
15  (add,2,2,2)
16  (div,3,1,3)
17  (goto,_,_,9)
18  (output,4,_,_)
19  (halt,_,_,_)
```

6. Convert BDIM with strings to BDIM using code.

```
 0  (16,0,37,0)
 1  (16,2,37,1)
 2  (1,37,37,2)
 3  (1,37,37,3)
 4  (16,0,37,4)
 5  (12,3,0,5)
 6  (13,5,37,9)
 7  (7,0,2,2)
 8  (7,0,3,3)
 9  (11,3,0,5)
10  (13,5,37,18)
11  (10,3,1,5)
12  (11,5,0,5)
13  (13,5,37,15)
14  (6,4,2,4)
15  (6,2,2,2)
```

```
16  (9,3,1,3)
17  (14,37,37,9)
18  (15,4,37,37)
19  (0,37,37,37)
```