Global CSE

Original Decaf:

```
import printf;

int g() {
    int a;
    int b;
    int c;
    int d;
    int e;
    int f;
    a = 1;
    b = 2;
    c = 3;
    e = a + b;
    d = a + b + c + e;
    f = (a + b) * e;
    printf("%d\n", f);
    return d;
}
void main() {
    int h;
    h = g();
    printf("%d\n", h);
}
```

Three Address Code list before

```
main: -> void {
        h: int = const 0
        a: int = const 0
        b: int = const 0
        c: int = const 0
        d: int = const 0
        e: int = const 0
        f: int = const 0
        a: int = const 1
        a: int = load a
        b: int = const 2
        b: int = load b
        c: int = const 3
        c: int = load c
        e: int = a + b          # a + b
        e: int = load e
        %0: int = load e
        %1: int = %0 + c           # a + b + c
        d: int = %1 + e           # a + b + c + e
        d: int = load d
        %2: int = load %0
        f: int = %2 * e           # (a + b) * e
        f: int = load f
        push f
        push @.string_0
        call int @printf           #  printf("%d\n", f)
        h: int = load d
        h: int = load h
        push h
        push @.string_0
        call int @printf           #  printf("%d\n", h)
        goto exit_main
    exit_main:
}
```

Three Address Code list after:

```
main: -> void {
        e: int = const 1 + const 2           # a + b
        %1: int = e + const 3           # a + b + c
        d: int = %1 + e           # a + b + c + e
        f: int = e * e           # (a + b) * e
        push f
        push @.string_0
        call int @printf           #  printf("%d\n", f)
        push d
        push @.string_0
        call int @printf           #  printf("%d\n", h)
        goto exit_main
    exit_main:
}
```

Copy Propagation

Original decaf code:

```
import printf;
int get_int ( int x ) {
  return x;
}
void main ( ) {
  int a, c, d;
  a = get_int ( 2 );
  c = 0;
  d = 0;
  printf ( "%d\n", a );
  printf ( "%d\n", c );
  printf ( "%d\n", d );
  c = a;
  a = get_int ( 3 );
  d = c;
  printf ( "%d\n", a );
  printf ( "%d\n", c );
  printf ( "%d\n", d );
}
```

Three Address Code list before:

```
main: -> void {
        a: int = const 0
        c: int = const 0
        d: int = const 0
        push const 2
        a: int = call @get_int          # get_int(2)
        a: int = load a
        c: int = const 0
        c: int = load c
        d: int = const 0
        d: int = load d
        push a
        push @.string_0
        call int @printf          #  printf("%d\n", a)
        push c
        push @.string_0
        call int @printf          #  printf("%d\n", c)
        push d
        push @.string_0
        call int @printf          #  printf("%d\n", d)
        c: int = load a
        push const 3
        a: int = call @get_int          # get_int(3)
        a: int = load a
        d: int = load c
        push a
        push @.string_0
        call int @printf          #  printf("%d\n", a)
        push c
        push @.string_0
        call int @printf          #  printf("%d\n", c)
        push d
        push @.string_0
        call int @printf          #  printf("%d\n", d)
        goto exit_main
    exit_main:
}
```

Three Address Code list after:

```
main: -> void {
        a: int = const 2
        c: int = const 0
        d: int = const 0
        push const 2
        push @.string_0
        call int @printf          #  printf("%d\n", a)
        push const 0
        push @.string_0
        call int @printf          #  printf("%d\n", c)
        push const 0
        push @.string_0
        call int @printf          #  printf("%d\n", d)
        c: int = const 2
        a: int = const 3
        push const 3
        push @.string_0
        call int @printf          #  printf("%d\n", a)
        push const 2
        push @.string_0
        call int @printf          #  printf("%d\n", c)
        push const 2
        push @.string_0
        call int @printf          #  printf("%d\n", d)
}
```

Dead Code Elimination and Dead Store Elimination

Original decaf code:
Three Address Code list before:

```
@.string_0 = "%d\n"        # 3 bytes

main: -> void {
        a: int = const 0
        b: int = const 0
        c: int = const 0
        d: int = const 0
        e: int = const 0
        a: int = const 0
        a: int = load a
        %0: int = const 2
        %1: int = a * %0          # a * 2
        %2: int = const 56
        b: int = %1 + %2          # a * 2 + 56
        b: int = load b
        %3: int = const 32
        %4: int = %3 * b          # 32 * b
        %5: int = const 25
        %6: int = a * %5          # a * 25
        c: int = %4 + %6          # 32 * b + a * 25
        c: int = load c
        %7: int = a * b          # a * b
        d: int = %7 % c          # (a * b) % c
        d: int = load d
        e: int = const 5
        e: int = load e
        push e
        push @.string_0
        call int @printf          #  printf("%d\n", e)
        goto exit_main
    exit_main:
}
```

Three Address Code list after:

```
.data
string_0:
      .string   "%d\n"
      .align 16
.text
.globl main
main:
      movq  $5, %rsi
      movq  $string_0, %rdi
      xorl  %eax, %eax
      callq printf
      jmp   .exit_main
.exit_main:
      xorl  %eax, %eax
      ret
```

Algebraic Simplification

Original decaf code:

```
import printf;

void main() {
    int a;
    a = 5 + 0;
    a = a / 1;
    a = a - 0;
    a = a * 1;
    printf("%d\n", a);
}
```

Three Address Code list before:

```
@.string_0 = "%d\n"        # 3 bytes

main: -> void {
        a: int = const 0
        %0: int = const 5
        %1: int = const 0
        a: int = %0 + %1          # 5 + 0
        a: int = load a
        %2: int = const 1
        a: int = a / %2          # a / 1
        a: int = load a
        %3: int = const 0
        a: int = a - %3          # a - 0
        a: int = load a
        %4: int = const 1
        a: int = a * %4          # a * 1
        a: int = load a
        push a
        push @.string_0
        call int @printf          #  printf("%d\n", a)
        goto exit_main
    exit_main:
}
```

Three Address Code list after:

```
.data
string_0:
      .string   "%d\n"
      .align 16
.text
.globl main
main:
      movq  $5, %rsi
      movq  $string_0, %rdi
      xorl  %eax, %eax
      callq printf
      jmp   .exit_main
.exit_main:
      xorl  %eax, %eax
      ret
```

Unreachable Code Elimination

Original decaf code:

```
import printf;

void main() {
    int a;
    bool b;
    b = true;
    if (b) {
        a = 59;
    } else {
        a = 5;
    }
    printf("%d\n", a);
}
```

Three Address Code list before:

```
@.string_0 = "%d\n"        # 3 bytes

main: -> void {
        a: int = const 0
        b: int = const 0
        b: bool = const 1
        b: bool = load b
    L0:
        if false b goto L1          #  if !(b)
        a: int = const 59
        a: int = load a
        push a
        push @.string_0
        call int @printf          #  printf("%d\n", a)
        goto exit_main
    L1:
        a: int = const 5
        a: int = load a
        push a
        push @.string_0
        call int @printf          #  printf("%d\n", a)
        goto exit_main
    exit_main:
}
```

Three Address Code list after:

```
main: -> void {
        push const 59
        push @.string_0
        call int @printf          #  printf("%d\n", a)
        goto exit_main
    exit_main:
}
```

Constant Propagation

Original decaf code:

```
int g() {
    int a;
    int b;
    int c;
    bool d;
    d = true;
    b = 10;
    if (d) {
        c = 5;
        a = 2;
        c = a + b;
    } else {
        c = 6;
        a = 7;
        c = a + b;
    }
    return c;
}
void main() {
    int b;
    b = g();
}
```

Three Address Code list before:

```
g: -> int {
        a: int = const 0
        b: int = const 0
        c: int = const 0
        d: int = const 0
        d: bool = const 1
        d: bool = load d
        b: int = const 10
        b: int = load b
    L0:
        if false d goto L1          #  if !(d)
        c: int = const 5
        c: int = load c
        a: int = const 2
        a: int = load a
        c: int = a + b          # a + b
        c: int = load c
        return c
        goto exit_g
    L1:
        c: int = const 6
        c: int = load c
        a: int = const 7
        a: int = load a
        c: int = a + b          # a + b
        c: int = load c
        return c
        goto exit_g
    exit_g:
}
```

Three Address Code list after:

```
g: -> int {
        a: int = const 0
        c: int = const 0
        d: bool = const 1
        b: int = const 10
    L0:
        if false const 1 goto L1          #  if !(d)
        a: int = const 2
        c: int = const 2 + const 10          # a + b
        return c
        goto exit_g
    L1:
        a: int = const 7
        c: int = const 7 + const 10          # a + b
        return c
        goto exit_g
    exit_g:
}
```