参考答案（0423）

- **1. 考虑右边的三地址代码：**

  (1) 把这段代码序列划分为基本块

```
(1)  s = 0
(2)  i = 0
(3)  t1 = s % 19
(4)  if t1 == 1 goto (19)
(5)  j = 0
(6)  if i < 50 goto (8)
(7)  return s
(8)  if j >= 100 goto (17)
(9)  if i == j goto (15)
(10) t2 = 100 * i
(11) t3 = t2 + j
(12) t4 = 4 * t3
(13) t5 = a[t4]
(14) s = s + t5
(15) j = j + 1
(16) goto (8)
(17) i = i + 1
(18) goto (3)
(19) return s
```

```
(1)  s = 0
(2)  i = 0
```

```
(3)  t1 = s % 19
(4)  if t1 == 1 goto (19)
```

```
(5)  j = 0
(6)  if i < 50 goto (8)
```

```
(7)  return s
```

```
(8)  if j >= 100 goto (17)
```

```
(9)  if i == j goto (15)
```

```
(10) t2 = 100 * i
(11) t3 = t2 + j
(12) t4 = 4 * t3
(13) t5 = a[t4]
(14) s = s + t5
```

```
(15) j = j + 1
(16) goto (8)
```

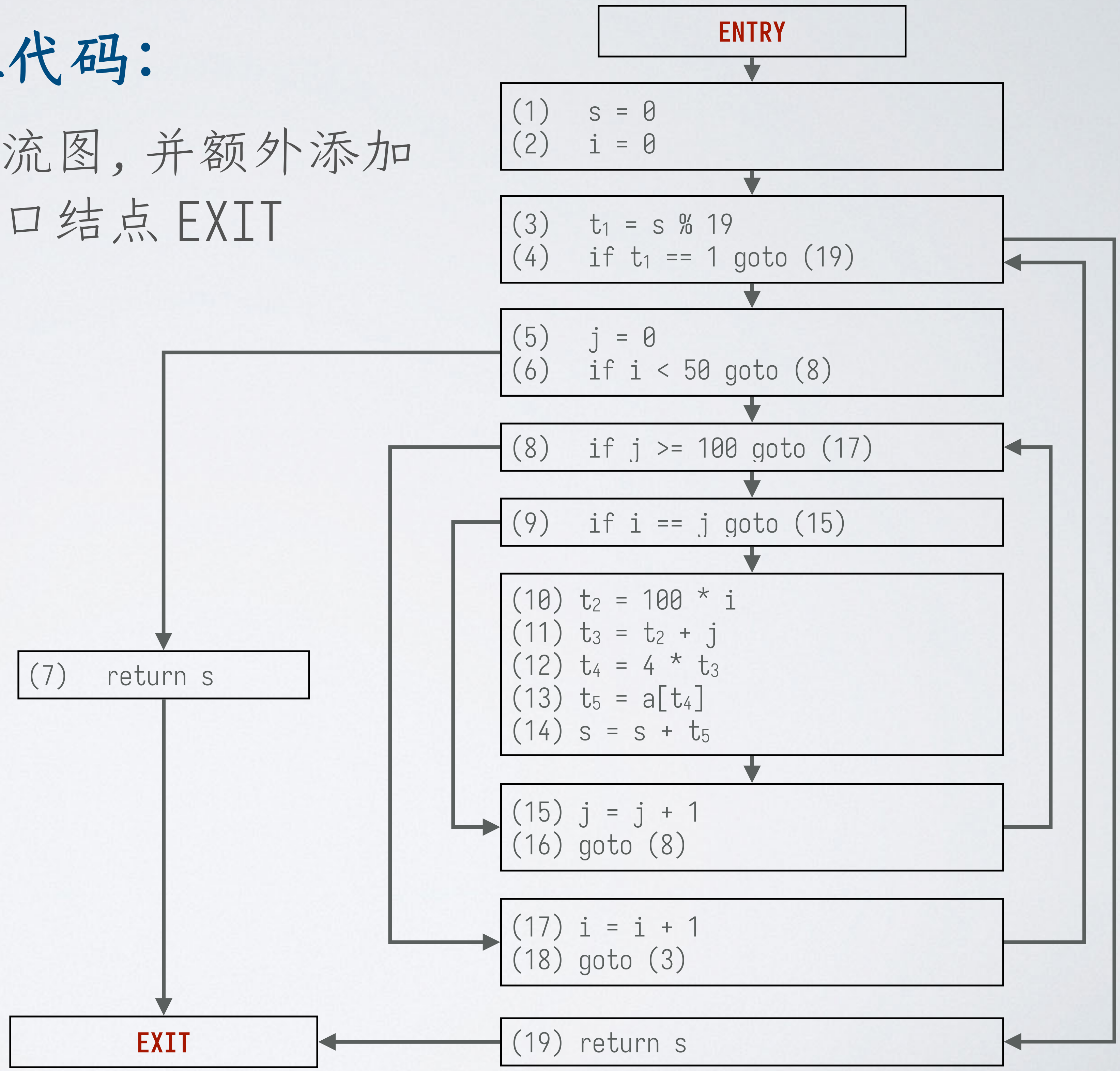
```
(17) i = i + 1
(18) goto (3)
```

```
(19) return s
```

参考答案（0423）

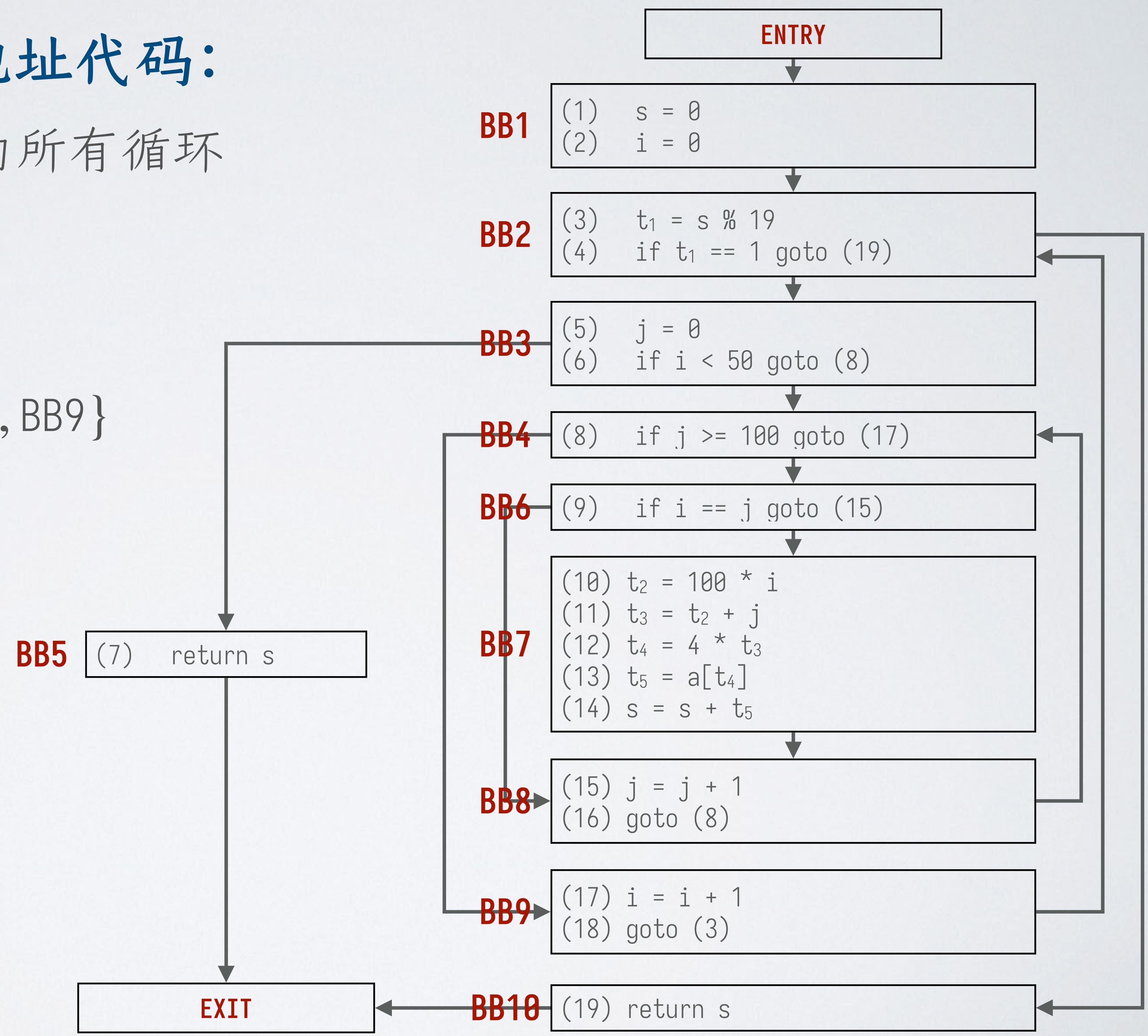
- **1. 考虑右边的三地址代码：**

  (2) 为这段代码构造控制流图，并额外添加入口结点 ENTRY 和出口结点 EXIT



参考答案（0423）

- **1. 考虑右边的三地址代码：**
  (3) 找出控制流图中的所有循环



循环一：

{BB2, BB3, BB4, BB6, BB7, BB8, BB9}

循环二：

{BB4, BB6, BB7, BB8}

参考答案（0423）

- 2. 按照课堂上给出的数组引用翻译的 SDT 给出下面语句 S 的三地址代码：

```
a[i*2+j][a[2][i+j][k]+i][i*4+k*2] = i*j+k*a[0][4][2*k];
```

其中数组 a 的类型是 int[5][6][9]。

```
t1 = i * 2
t2 = t1 + j
t3 = i + j
t4 = 2 * 6
t5 = t4 + t3
t6 = t5 * 9
t7 = t6 + k
t8 = t7 * 4
t9 = a [ t8 ]
t10 = t9 + i
t11 = t2 * 6
t12 = t11 + t10
t13 = i * 4
t14 = k * 2
t15 = t13 + t14
```

```
t16 = t12 * 9
t17 = t16 + t15
t18 = t17 * 4
t19 = i * j
t20 = 0 * 6
t21 = t20 + 4
t22 = 2 * k
t23 = t21 * 9
t24 = t23 + t22
t25 = t24 * 4
t26 = a [ t25 ]
t27 = k * t26
t28 = t19 + t27
a [ t18 ] = t28
```

参考答案（0423）

- 3. 按照课堂上给出的控制流回填翻译的 SDT 给出下面语句 S 的三地址代码和 S.nextlist (指令编号从 100 开始):

```
while ((a + b * c > x + y) && (m == n)) {
  if (x <= y) {
    while (a < b) { a = a + 10; b = c * m; }
  } else a = b + c;
}
```

```
100: t1 = b * c
101: t2 = a + t1
102: t3 = x + y
103: if t2 > t3 goto 105
104: goto _
105: if m == n goto 107
106: goto _
107: if x <= y goto 109
108: goto 117
109: if a < b goto 111
```

```
110: goto 116
111: t4 = a + 10
112: a = t4
113: t5 = c * m
114: b = t5
115: goto 109
116: goto 119
117: t6 = b + c
118: a = t6
119: goto 100
```

S.nextlist $= \{104, 106\}$

参考答案（0423）

- 4. 考虑一种基于栈的中间表示，它的所有指令都隐式地操作一个全局的栈，填写以下生成短路求值代码的 SDT。

产生规则	语义动作
$S \rightarrow$ ID = E ;	{ S.code = E.code ‖ `SET_VAR({genvar(ID.lexeme})`; }
$E \rightarrow E_1 + E_2$	{ E.code = E_1.code ‖ E_2.code ‖ `ADD`; }
$E \rightarrow - E_1$	{ E.code = `CONST(0)` ‖ E_1.code ‖ `SUB`; }
$E \rightarrow$ ID	{ E.code = `GET_VAR({genvar(ID.lexeme)})`; }
$B \rightarrow$ true	{ B.code = `GOTO({$B$.true})`; }
$B \rightarrow E_1 == E_2$	{ B.code = E_1.code ‖ E_2.code ‖ `SUB` ‖ `GOTO_IF_ZERO({$B$.true})` ‖ `GOTO({$B$.false})`; }
$B \rightarrow$ B_1 || B_2	{ B_1.true = B.true; B_1.false = genlabel(); } { B_2.true = B.true; B_2.false = B.false; } { B.code = B_1.code ‖ `{$B_1$.false}:` ‖ B_2.code; }
$S \rightarrow$ if (B) S_1 else S_2	{ B.true = genlabel(); B.false = genlabel(); } { S_1.next = S.next; } { S_2.next = S.next; } { S.code = B.code ‖ `{$B$.true}:` ‖ S_1.code ‖ `GOTO({$S$.next})` ‖ `{$B$.false}:` ‖ S_2.code; }
$S \rightarrow$ while (B) S_1	{ B.true = genlabel(); B.false = S.next; } { S_1.next = genlabel(); } { S.code = `{$S_1$.next}:` ‖ B.code ‖ `{$B$.true}:` ‖ S_1.code ‖ `goto {$S_1$.next}`; }