Компилятор для языка COREC

Язык COREC (COmputed RECurrences)

- Это **"язык для специфической области"** (**Domain Specific Language, DSL**).
- Он предназначен для **упрощения программирования вычислений на многомерных массивах**.
- Все вычисления определяются через **рекуррентные зависимости**.

Структура программы на языке COREC

```
prog MonProg {

  def Fonc1 { …
  }

  def Fonc2 { …
  }

  ...
  def Main { …
  }

}
```

Функции в языке COREC

```
def Fonc {

/* аргументы */

...

/* локальные переменные */

Loc: …

/* секция Dom */

Dom: …

/* секция Rec */
```

```
Rec: …

}
```

Аргументы функций

```
def Fonc {

/* аргументы */

...

/* локальные переменные */

Loc: …

/* секция Dom */

Dom: …

/* секция Rec */

Rec: …

}
```

- Аргументы входа:

  ```
  in: список_аргументов
  ```
- Аргументы выхода:

  ```
  out: список_аргументов
  ```
- Аргументы входа/выхода:

  ```
  inout: список_аргументов
  ```

Аргументы функций

```
def Fonc {

/* аргументы */

...

/* локальные переменные */

Loc: …

/* секция Dom */
```

```
Dom: …

/* секция Rec */

Rec: …

}
```

- Массивы: неявно имеют тип "float".
- Простые переменные: целые числа или числа с плавающей запятой, но переменные с плавающей запятой должны быть объявлены как массивы размером 1.
- Главная функция `Main` не принимает аргументов.
- Аргумент из списка `out:`:
  - Может появляться в списках `in:` или `inout:` других функций.
  - Не может появляться в списке `out:` другой функции.
  - Его область видимости определяется вызывающими функциями.
  - Одно и то же имя должно использоваться всеми другими функциями, которые имеют доступ к этому аргументу.

Доступ функций

- Функция имеет доступ только к:
  - своим аргументам,
  - своим локальным переменным,
  - аргументам выхода функций, которые она вызывает.
- Функция может вызывать другую функцию только в том случае, если она может предоставить ей аргументы `in:` и `inout:`.

  ```
  ⇒ последовательность вызовов функций должна учитывать
  цепочку зависимостей между функциями!
  ```

- Возвращаемое значение функции:
  - представляет собой набор её аргументов `out:` и `inout:`.
  - может быть присвоено простой переменной только в том случае, если возвращаемое значение уникально и является простой переменной.

Пример программы на COREC

```
prog doitgen {
```

```
  def Inita {
    [in: NR,NQ,NP; out: (a,NR,NQ,NP)]
    …
  }

  def Initc4 {
    [in: NP; out: (c4,NP,NP)]
    …
  }

  def Eq2 {
    [in: (c4,NP,NP), NR,NQ,NP; inout: (a,NR,NQ,NP)]
    …
  }

  def Main {
    …
    print(Eq2(Initc4,Inita))
    …
  }

}
```

Локальные переменные

```
def Fonc {

/* аргументы */

...

/* локальные переменные */

Loc: …

/* секция Dom */

Dom: …

/* секция Rec */

Rec: …

}
```

- Локальные переменные:
  - Опциональны.
  - Могут быть массивами (типа `float`) или простыми переменными

(целые числа или числа с плавающей запятой). Однако числа с плавающей запятой объявляются как массивы размером 1.
- Могут быть использованы как аргументы входа (`in:`) или входа/выхода (`inout:`) для вызова других функций.

Секция Dom

```
def Fonc {

/* аргументы */

...

/* локальные переменные */

Loc: …

/* секция Dom */

Dom: …

/* секция Rec */

Rec: …

}
```

- Опциональна.
- Определяет **пространство рекуррентности**, то есть значения индексов, для которых выполняются вычисления, описанные в секции `Rec` .
- Значения задаются с использованием **интервалов**.
- Порядок объявления интервалов определяет порядок выполнения вычислений.

Пример:

```
Dom: i in [0..NR-1], j in [0..NQ-1], k in [0..NP-1]
```

Эквивалентно:

```
for (i=0; i<NR; i++)
  for (j=0; j<NQ; j++)
    for (k=0; k<NP; k++)
      …
```

- Границы интервалов:

- Могут быть целыми константами, целыми переменными, индексами рекуррентности, ранее определёнными, или арифметическими выражениями.

Пример:

```
Dom: i in [0..N-1], k in [0..i-2]
```

Секция Rec

```
def Fonc {

/* аргументы */

...

/* локальные переменные */

Loc: …

/* секция Dom */

Dom: …

/* секция Rec */

Rec: …

}
```

- Содержит инструкции функции:
  - Присваивания.
  - Условные инструкции.
  - Вызовы функций.
- Инструкции имеют доступ только к:
  - локальным переменным,
  - аргументам функции,
  - аргументам выхода функций, вызванных ранее.

Вызовы функций

1. Функциональная запись:

```
fonc(fonc1, fonc2, ..., fonci(fonci1, fonci2, ...), ...)
```

2. Ссылки на относительные вызовы:

```
fonci1
fonci2
fonc1(%2, %1)
fonc(%1)
```

3. Комбинированная запись:

```
fonci1
fonci2
fonc(fonc1(%2, %1))
```

- Порядок вызовов должен учитывать зависимости данных между вызовами.
- В функциональной записи вызывающая функция имеет доступ только к выходным данным последней вызванной функции.
- При последовательных вызовах с использованием относительных ссылок каждая выходная переменная становится доступной для вызывающей функции.

Предопределённые функции

- `read` : чтение значения простой переменной (целое или число с плавающей запятой) с клавиатуры.
- `print` : вывод на экран значения простой переменной, массива или выходных данных функций.
- `printstr` : вывод строки символов.

Условные инструкции

```
condition? instructions_si_vrai : instructions_si_faux
```

- Условие ограничено выражением сравнения двух операндов с использованием реляционного оператора.
- Операнды могут быть арифметическими выражениями, константами, простыми переменными или элементами массива.
- Логические операторы (и, или, ...) не поддерживаются.

Пример программы

```
prog doitgen {
```

```
  def Inita { [in: NR,NQ,NP; out: (a,NR,NQ,NP)]
    Dom: i in [0..NR-1], j in [0..NQ-1], k in [0..NP-1]
    Rec: a[i,j,k]=((i*j + k) % NP) / NP
  }

  def Initc4 { [in: NP; out: (c4,NP,NP)]
    Dom: i in [0..NP-1], j in [0..NP-1]
    Rec: c4[i,j]=(i*j % NP) / NP
  }

  def Eq1 { [in: (sum,NP),r,q,NR,NQ,NP; inout:
(a,NR,NQ,NP)]
    Dom: k in [0..NP-1]
    Rec: a[r,q,k] = sum[k]
  }

  def Eq2 { [in: (c4,NP,NP), NR,NQ,NP; inout:
(a,NR,NQ,NP)]
    Loc: r,q,
         (sum,NP)
    Dom: i in [0..NR-1], j in [0..NQ-1], k in [0..NP-1], l
in [0..NP-1]
    Rec: {
      l==0?sum[k]=0: ;
      sum[k] += a[i,j,l] * c4[l,k];
      k==NP-1?l==NP-1?
      { r=i;
        q=j;
        Eq1 }::
    }
  }

  def Main {
      Loc: NR,NQ,NP
      Rec: {
        printstr("Entrez une valeur entière NR : ");
        read(NR);
        printstr("Entrez une valeur entière NQ : ");
        read(NQ);
        printstr("Entrez une valeur entière NP : ");
        read(NP);
        printstr("Le résultat est : ");
        print(Eq2(Initc4,Inita))
      }
  }

}
```

Обнаружение ошибок

- Важно обеспечить подробные и понятные сообщения об ошибках, включая:
  - Неправильные зависимости между функциями.
  - Использование не определённых индексов в интервалах секции `Dom`.
  - Использование переменных или массивов, недоступных текущей функции.
  - Повторное использование имён переменных или массивов в нескольких выходных аргументах разных функций или в локальных переменных и аргументах одной функции.

**Реализация **

- Требуется генерация ассемблерного кода для **MIPS** или **RISC-V** (на выбор).
- Компилятор должен генерировать корректный код даже для ограниченного подмножества языка COREC.
- Функция `print` должна быть реализована обязательно для проверки результатов.