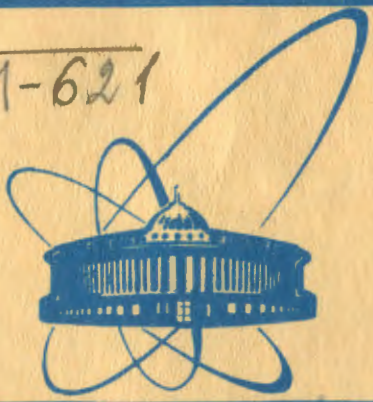

сообщения
объединенного
института
ядерных
исследований
дубна

11-81-671

Ли Рен Хи

ДВУМЕРНОЕ ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕР

1981

Введение

В практике программирования существуют два подхода к изображению программ:

- текстуальное изображение,
- графическое изображение.

Текстуальное изображение программ в значительной степени зависит от используемого языка программирования и по этой причине читабельность текста в значительной степени зависит от культуры записи, которой владеет программист, от того, насколько он уважает возможных читателей его текста. В силу разнообразия степени владения этой культурой, а также необходимости наличия самого общего представления о существе реализованного алгоритма в отдельной документации используется графическая форма изображения программ. При этой форме изображения обычно используются блок-схемы. Известно, что блок-схемы не позволяют адекватно изображать реализованный алгоритм, они только подсказывают читателю основную идею реализации.

О нетехнологичности описанного подхода в разработке программ и составлении отчетной документации хорошо сказано в книге И.В.Вельбицкого /I/.

Данная работа является развитием идей двумерного программирования, изложенных в /I/ в применении их к машинно-ориентированным языкам (автокоды, ассемблеры).

Напомним существующие различия нашего подхода двумерного изображения программ и подхода, реализованного в R -технологии И.В.Вельбицкого:

- R-технология в принципе рассчитана только на некую абстрактную RBM -машину, на которой реализован R -язык. По этой причине R -технология не может быть применена к другим средствам программирования.

В R-технологии двумерное изображение программы основано, как и у нас, на представлении программы ориентированным нагруженным графом.

В - R -технологии узлы графа изображают метки программы, наш подход ставит в соответствие каждому узлу графа одну булевскую процедуру -предикат.

В данной работе мы покажем, что наш метод изображения программ (который дает изображение, адекватное текстуальному представлению) можно применять не только к языкам типа CDL /2/, но и к машинно-ориентированным языкам.

Абстрактная машина

Допустим, что в нашем распоряжении имеется некоторая ЭВМ с набором команд, данных в следующей таблице.

Таблица I.

№	Мнемокод	Содержание операции
I	LOAD	Считывание слова из памяти
2	STORE	Запись слова в память
3	ADD	Сложение
4	SUB	Вычитание
5	BZ	Условный переход при нулевом содержимом сумматора
6	BNZ	Условный переход при нулевом содержимом сумматора
7	B	Безусловный переход
8	CALL	Безусловный переход с запоминанием адреса возврата
9	READ	Считывание символа с перфоленты в память
I0	PRINT	Печать слова из памяти
II	STOP	Стоп

С точки зрения наличия команд передачи управления можно дать этому набору команд следующую классификацию:
- линейные команды (LOAD, STORE, ADD, SUB, READ, PRINT, STOP);
- команды с условным переходом (BZ, BNZ);
- команды с безусловным переходом (B);
- команды вызова алгоритма (CALL).

Обозначения

В работе /2/ был дан метод двумерного изображения алгоритмов для языка CDL , в котором горизонтальное ребро графа обозначает набор операций алгоритма, если предшествующий предикат (который стоит в узле графа) дает результат: "Истина", а вертикальное ребро графа обозначает набор операций алгоритма, если предшествующий предикат дает результат: "Ложь". Такой способ построения графа мы называем YES - NO метод, а граф, соответствующий этому методу построения, называем YES - NO граф. Для машинно-ориентированного языка (назовем его просто ассемблер) применим метод NO - YES и, соответственно, NO - YES граф, как это станет далее видно из наших обозначений.

Мы предполагаем, что синтаксис ассемблера не имеет существенного значения в данной работе и что читатель, знакомый с машинно-ориентированными языками, поймет синтаксис записи команд ассемблера без дополнительных разъяснений.

Линейные команды на ребрах графа еще изображаем следующим образом:

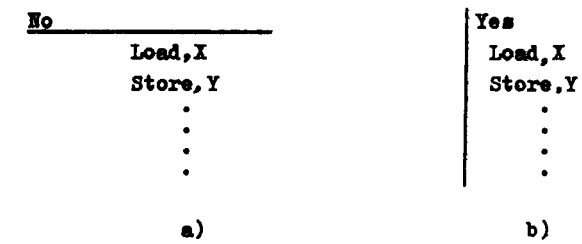

рис.I

Метки так:

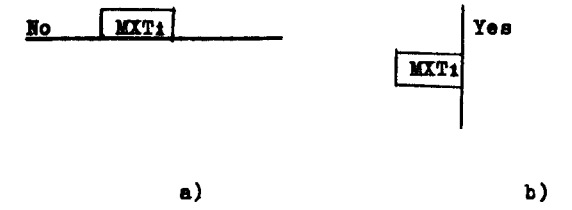

рис.2

Команды с условным переходом изображаются узлом графа:

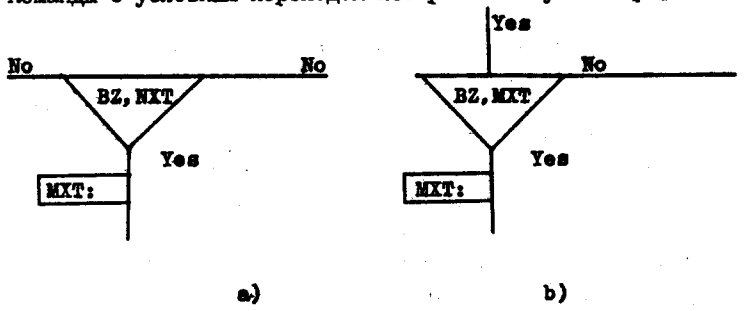

рис.3.

Команды с безусловным переходом:

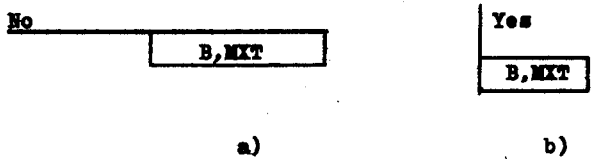

рис.4.

Команды вызова алгоритма:

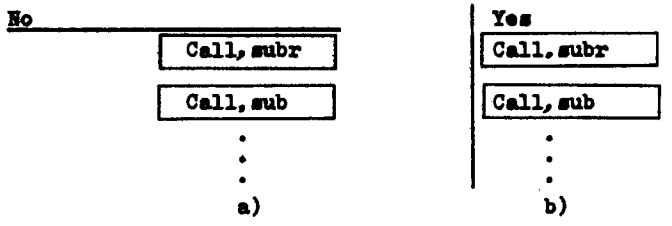

рис.5.

При реализации перехода по ключу существует фрагмент алгоритма, который содержит список всех возможных переходов по состоянию ключа. Этот фрагмент мы обозначаем так:

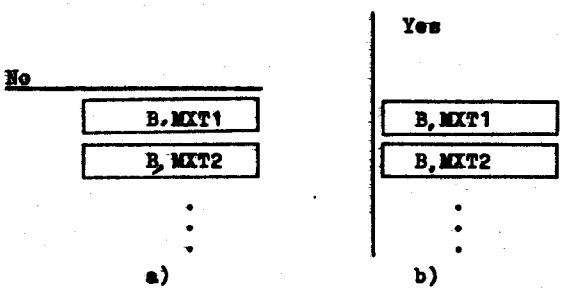

рис.6.

Связующие символы, которые мы используем тогда, когда на одном листе бумаги нельзя представить граф всего алгоритма:

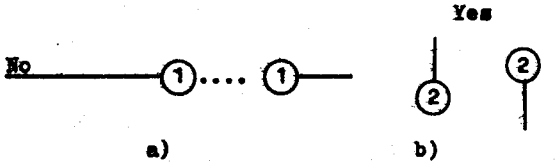

рис.7.

Правила двумерного программирования

Эти правила следует изучать после ознакомления с введенными выше обозначениями.

I) На бумаге программа начинается в левом крайнем углу сверху с горизонтального ребра (рис.9). Над этим ребром пишется заголовок программы. Если встречаются декларации, то они пишутся по этим ребрам как линейные команды.

2) Линейные команды (рис.I), команды вызова алгоритмов (рис.5) и команды безусловного перехода, фрагменты ключей (рис.6) в случае горизонтального ребра пишутся по порядку под текущим ребром, в случае вертикального ребра - справа от него.

3) Если метка находится на горизонтальном ребре, то она обозначается в четырехугольнике над ребром (рис.2а), на вертикальном ребре - слева от него.

4) Команда условного перехода обозначается треугольником (рис. 3), выход с результатом "Истина" (**Yes**) обозначается вертикальным ребром, а с результатом "Ложь" (**No**) - горизонтальным.

Правая метка условного перехода ставится налево от вертикального ребра (**Yes**) и заключается в четырехугольник.

5) Команда безусловного перехода на горизонтальном ребре обозначается внутри четырехугольника под этим ребром (рис.4а), а на вертикальном - обозначается аналогично, но справа от этого ребра (рис.4в).

6) Программа пишется слева направо. Если встретился узел, то прежде всего надо нагрузить горизонтальное ребро командами, которые будут выполняться в случае, если предикат дает результат "Ложь", затем нагрузить вертикальное ребро, если предикат дает результат "Истина".

7) Если горизонтальное ребро полностью нагружено, то надо начинать нагружать вертикальное ребро, которое находится на наиболее близкой стороне к последнему нагруженному горизонтальному ребру.

8) Команду безусловного перехода используют только при переходе на верхнюю часть (т.е. уже нагруженную часть графа).

Если в программе (графе программы) была описана структура с левой меткой, то данную метку нет смысла опять описывать на ребре графа. На рис.9 лишние ребра, которые содержат повторное описание левой метки, зачеркнуты.

При соблюдении пунктов 6,7,8 команды, передающие управление сверху-вниз, устраняются, что приводит к уменьшению общего числа команд.

9) При программировании ключа фрагмент, содержащий переходы, записывается в той последовательности, в которой это необходимо алгоритму.

10) Реконструкция текста программы с ее графического изображения происходит при движении по графу слева направо по часовой стрелке.

Пример.

Предположим, что имеется линейный массив длиной 100 ячеек памяти, содержащий нули и единицы. Необходимо произвести подсчет всех единиц в этом массиве и произвести печать.

Решение

Заведем константы:

C100 со значением "100";
C1 со значением "1";
C0 со значением "0";
- ячейку памяти **Position** будем использовать для нумерации ячейки буфера;
- **Count 1** будем использовать как счетчик единиц;
- A - для запоминания считанного кода.

Вначале напишем заголовок алгоритма и установим начальные значения **Position** и **Count 1** (см.рис.9).

Для организации цикла заведем левую метку **Cycle:**, далее прочитаем значение в A с одновременным уменьшением **Position** на "1". Делаем проверку **Position** на значение "0".

Если **Position** ≠ 0 , то проверяется значение A на нуль. Если A≠0, то значение **Count 1** увеличиваем на "1" и передаем управление на метку **Cycle** , в противном случае состояние **Count 1** не меняется, а управление передается опять на метку **Cycle** . Если **Position=0**, то управление передается на метку **Finish** , после чего, если A≠0, то значение **Count 1** увеличивается на "1" и печатается содержимое **Count 1** с последующим остановом действия алгоритма. Если A=0, то производится печать счетчика единиц **Count 1** с окончанием действия алгоритма.

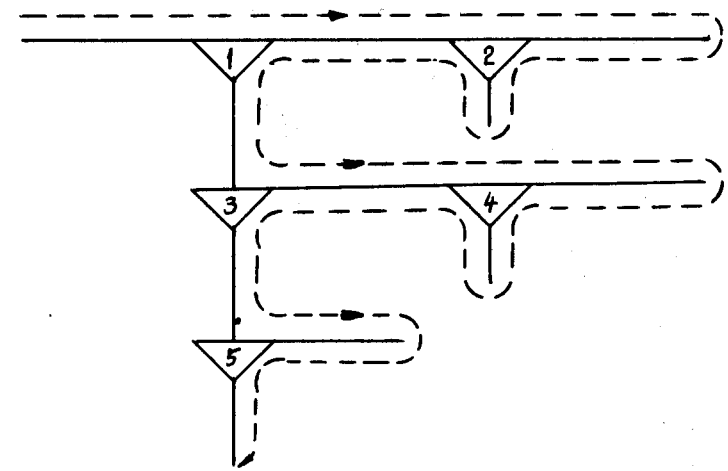

рис.8.

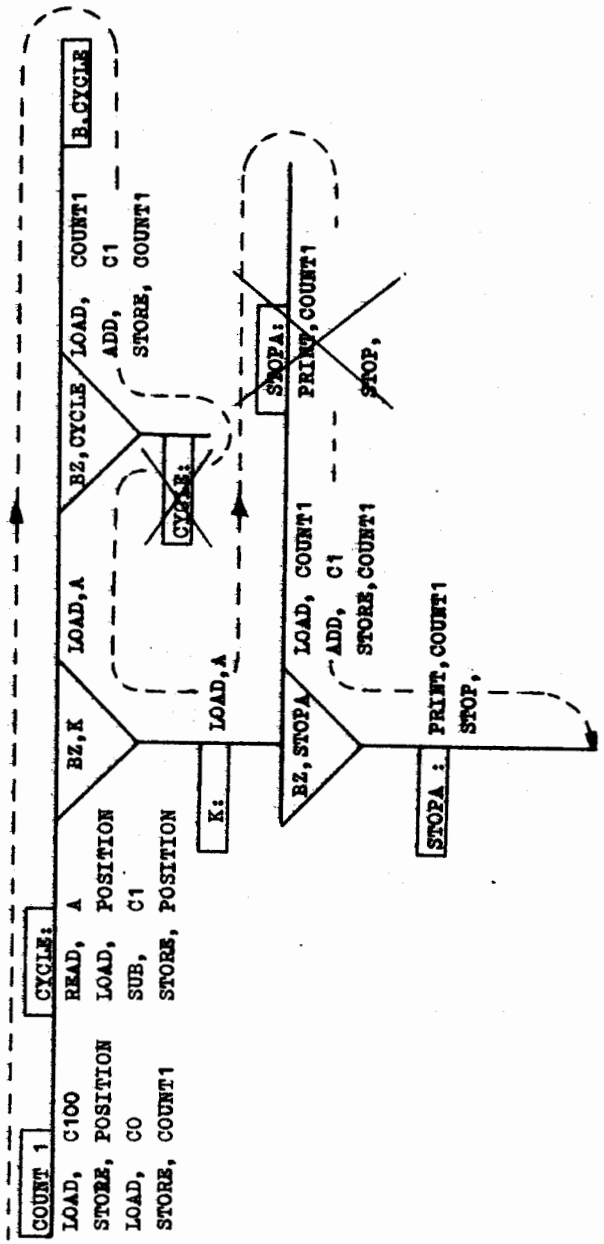

Рис.9

Метки **Cycle, Stop A** на рис.9 появляются несколько раз, повторное их объявление зачеркнуто.

На рис.10 показана текстуальная запись данного алгоритма, восстановленная по ранее описанной схеме.

```
COUNT1
        LOAD  , C100
        STORE, POSITION
        LOAD  , CO
        STORE, COUNT1
CYCLE:  READ  , A
        LOAD  , POSITION
        SUB   , C1
        STORE, POSITION
        BZ    , K
        LOAD  , A
        BZ    , CYCLE
        LOAD  , COUNT1
        ADD   , C1
        STORE, COUNT1
        B     , CYCLE
K:      LOAD  , A
        BZ    , STOPA
        LOAD  , COUNT1
        ADD   , C1
        STORE, COUNT1
STOPA:  PRINT, COUNT1
        STOP  ,

C100:          100
C1   :           1
CO   :           0
POSITION:
COUNT1  :
A       :
```

Рис.10.

Заключение

Перечислим преимущества описанного метода программирования в сравнении с другими способами:

- двумерное изображение программы полностью отображает логику алгоритма и может служить как начальным, так и конечным документом;
- в процессе эксплуатации визуально проще находить места корректировки и проводить необходимые исправления;
- перекодировка двумерного изображения алгоритма в текстуальную форму может проводиться механически лаборантом;
- ошибки в процессе составления программы встречаются гораздо реже, чем при текстуальном методе программирования;
- двумерное изображение программы является полным, исчерпывающим документом для возможных его читателей.

В заключение автор выражает искреннюю благодарность А.А.Хошенко и Р.И.Гайдамаке за обсуждения и помощь в подготовке рукописи данной работы к печати.

Литература

1. Вельбицкий И.В., Ходаковский В.Н., Шолмов Л.Н. Технологический комплекс производства программ на машинах ЕС ЭВМ БЭСМ-6, М., „Статистика", 1980.

2. Ли Рен Хи, Хошенко А.А. ОИЯИ, Р11-80-804, Дубна, 1980.

3. **Berztiss A.T. Data Structures Theory and Practice, University on Pittsburgh, 1971 .**

Рукопись поступила в издательский отдел
27 октября 1981 года.