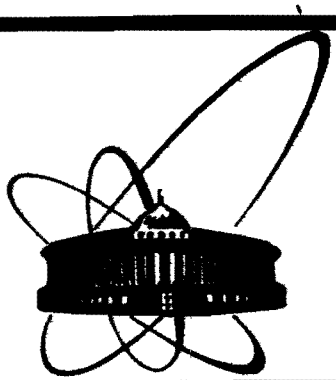

сообщения
объединенного
института
ядерных
исследований
дубна

Р11-86-172

В.В.Вицев, Л.А.Егошин, А.К.Ломов

КОМПИЛЯТОР И БИБЛИОТЕКИ ЯЗЫКА ФОРТРАН-4 В ОПЕРАЦИОННОЙ СИСТЕМЕ UNIX ДЛЯ ЭВМ СМ-4

1986

ВВЕДЕНИЕ

В ОС UNIX/1/ для ЭВМ СМ-4 используется компилятор языка фортран ОС UNIX V6. Этот компилятор соответствует стандарту ANSI-66. Однако некоторые особенности компиляции /только по одной подпрограмме в файле и невозможность использования 8-битного кода в операторах format/ затрудняют его использование.

Серьезным недостатком является крайне низкая скорость работы создаваемых программ. Это объясняется тем, что компилятор использует библиотеки, рассчитанные на работу с FPP, которого нет на ЭВМ СМ-4, а эмуляция команд значительно снижает скорость работы.

Была поставлена задача добиться устранения указанных недостатков. С этой целью написан препроцессор, который разбивает исходный файл на отдельные подпрограммы, выполняет необходимые преобразования над ними и передает их на дальнейшую обработку компилятору. Ускорение работы достигается за счет изменения алгоритмической структуры двух библиотек фортрана - библиотеки интерпретации и библиотеки математических функций. Модернизированная нами структура учитывает особенности команд процессора СМ-4, в результате чего скорость работы стандартных функций фортрана /синус, экспонента, корень и др./ в отдельных случаях увеличивается в 100 раз.

1. РАБОТА С КОМПИЛЯТОРОМ

1.1. Вызов компилятора

ftn [параметры] файлы

Компилятор обрабатывает несколько типов исходных файлов:

*.F - тексты на языке фортран-4;
*.f - тексты на входном языке компилятора /по одной подпрограмме в файле и т.д./;
*.s - тексты на языке ассемблер/3/.

Параметры, обрабатываемые компилятором:

-f - сохранение файлов *.f, созданных препроцессором;
-s - вывод результатов компиляции в файлы *.s, объектные файлы не создаются;
-c - только создание объектных файлов *.o;
-v - запрет вывода на терминал имен компилируемых файлов;
-d - диагностика работы ftn.

Остальные аргументы рассматриваются как параметры редактора связей, либо как объектные файлы, либо как объектные библиотеки. Вместе с результатами компиляции они используются для создания выполняемого файла a.out.

1.2. Этапы работы ftn

ftn имеет следующие этапы работы:

1/ Препроцессор
- Вход - файл с исходным текстом на фортране-4.
- Выход - файлы с текстом на входном языке компилятора.
- Программа - /usr/bin/ftn

2/ Компилятор
- Вход - файл с текстом на входном языке компилятора.
- Выход - файл с ассемблерной программой.
- Программа - /usr/fort/fc1

3/ Ассемблер
- Вход - файл с ассемблерной программой.
- Выход - объектный файл.
- Программа - /bin/as.

4/ Редактор связей
- Вход - объектные файлы и объектные библиотеки.
- Выход - файл a.out с выполняемой программой.
- Программа: - /bin/ld

1.3. Препроцессор

Препроцессор, который вызывается командой ftn, выполняет ряд функций, а именно:

- запускает компиляцию программы, написанной на фортране-4;
- выполняет компиляцию исходного файла, состоящего из многих подпрограмм, разделяя их по отдельным файлам с именами либо MAIN.f, либо blockdata?.f, либо <имя подпрограммы> .f, поочередно пропуская их через компилятор и собирая позже в одну выполняемую программу;
- воспринимает стандартную форму карт продолжения.

Препроцессор работает, создавая промежуточные файлы, которые уничтожаются в конце работы. Средняя скорость обработки препроцессора - около 500 строк в секунду.

1.4. Компилятор

В компиляторе сделаны два изменения: введен восьмибитный код /так как в ОС типа UNIX-ИНМОС и.т.д. принят восьмибитный код, то при этом для кодирования русских букв используется старший бит/ и заменен системный вызов seek на lseek для ОС UNIX V7.

1.5. Особенности входного языка

Имеется ряд особенностей входного языка, которые необходимо учитывать при переносе программного обеспечения:

- нет оператора backspace;
- не реализованы оператор-функции;
- не реализован масштабный множитель на вводе;
- служебные слова воспринимаются только строчными буквами;
- в операторе data каждый инициализируемый элемент массива нужно указывать явно, но символьная строка может инициализировать весь массив или последовательность его элементов;
- при форматном выводе первый символ не является управляющим, поскольку вывод может быть направлен как на печать, так и на дисплей;
- ввод-вывод поддерживает до 20 номеров каналов /0-19/; при этом канал 5 является стандартным вводом, канал 6 - стандартным выводом, а остальные каналы работают с файлами fortnn в текущей директории;
- есть оператор implicit;
- есть дополнительные типы данных: double complex, logical*1, integer*1, integer*2;
- холлеритовские константы можно задавать, ограничивая их символами одиночной и двойной кавычки;
- при форматном вводе данные можно задавать в сводобном формате, перечисляя их через запятую /для форматов I,F,E,D,G и L/;
- пробел и символ табуляции эквивалентны;
- оператор может начинаться с любой позиции;
- добавлен оператор include файл, позволяющий вставлять содержимое указанного файла в текст.

2. ВНУТРЕННЯЯ СТРУКТУРА ПРОГРАММ

2.1. Внутренний формат подпрограмм

Компилятор выдает оттранслированную программу в виде последовательности адресов подпрограмм из библиотеки интерпретации filib.a и их аргументов. Это - так называемый "сшитый"/threaded/ код. Вывод результатов трансляции оформляется в виде ассемблерной программы /см.п.2.4/. Принцип организации выполнения в виде обращения к библиотеке подпрограмм широко распространен для минимашин с их упрощенной системой команд. Набор этих подпрограмм можно рассматривать как команды некой виртуальной машины. Такой подход дает менее быстрые программы, чем генерация прямых кодов /inline или native code/, но обладает компактной структурой самого компилятора и генерируемых им программ. Данный компилятор не выполняет оптимизацию кода, поэтому программистам рекомендуется использовать общепринятые правила оптимизации фортрановских программ /7/.

2.2 Распределение памяти под программу

Память под программу разбивается на три сегмента и стек. В текстовом сегменте размещены сама программа и константы. Переменные, инициализируемые оператором data, помещаются в инициализированный сегмент данных - data. Неинициализируемые переменные размещаются в сегменте bss. В этом сегменте также размещаются переменные из списка common. При хранении программы на диске место для сегмента не отводится. По принятому в ОС UNIX правилу эта область заполняется нулями перед началом работы, а потому все неинициализированные переменные имеют начальное значение, равное нулю.

На этапе сборки программы сначала собираются все текстовые сегменты разных подпрограмм и библиотечных функций, затем все сегменты data и, наконец, определяется размер области bss, суммарно равный общему объему всех сегментов bss. Общий размер всех сегментов не должен превышать 64 Кбайта. Распределение памяти при выполнении фортрановских программ приведено в табл.1 /адреса виртуальные/.

Таблица 1

000000		text
	Пролог - начало программы на фортране: mov sp,argp /сохранение значения sp mov $main,r4 /переход на главную программу jmp *(r4)+ /	
	rerr - подпрограмма обработки ошибок	
	temp - область возвращаемых значений для стандартных функций. Это модифицируемая часть сегмента•text	
	Текстовая часть главной /main/ программы, функций и подпрограмм	
	Модули стандартных функций фортрана из библиотеки libf.a	
	Модули библиотеки интерпретации filib.a	
	Данные из операторов data	•data
	Блоки common и переменные из подпрограмм и функций фортрана	•bss
177776	Стек	

2.3. Использование регистров

Во время выполнения программы регистры r3 и r4 имеют специальное назначение: r3 используется в функциях и подпрограммах как указатель на область аргументов, а r4 является аналогом регистра - счетчика команд /в виртуальной машине/. В большинстве случаев r4 содержит адрес памяти, из которой извлекаются адрес следующей исполняемой подпрограммы библиотеки интерпретации, переход на которую осуществляется с помощью команды jmp *(r4)+, завершающей выполнение подпрограммы. На использование остальных регистров, кроме sp и pc, не накладывается никаких ограничений. В силу этого программист может изменять значение регистров r0, r1, r2 и r5 по своему усмотрению.

2.4. Обращение к подпрограммам

Имеются следующие особенности в структуре вызова и возврата:

- подпрограмме или функции передается адрес аргумента;
- список адресов аргументов формируется каждый раз перед вызовом;
- если аргументы являются выражениями, то их значения помещаются в стек;
- есть функции, работающие с переменным числом аргументов, поэтому указывается число переданных аргументов;
- возвращаемый функцией результат может иметь различный тип.

Применяемый алгоритм связи подпрограмм учитывает эти особенности. Ниже приведены примеры генерации кода компилятором для главной программы и подпрограммы-функции.

Программа:

```
      real a, b, c
      a=b
      b=c(a)
      end
```

Ассемблерный код:

```
main: rval4; b_     /запись в стек значения b длиной 4 байта;
      gmv4; a_      /запись в a значения из стека;
                    /формирование списка адресов аргументов;

      stsp; oa      /сохранить значение стека до вызова;
      lval; a_      /адрес a записать в стек;
      svst; oa+2    /запись адреса аргумента
                    /    в область аргументов (oa);
      call; c.; oa; 1.; 4.
                    /вызов подпрограммы call с параметрами:
                    /
                    /адрес функции,
                    /адрес oa,
                    /число аргументов,
                    /длина результата.
```

```
        gmv4; b_    /запись в b результата из стека;
        stop; 0     /завершение программы;
.even
.bss                /сегмент    неинициализируемых данных;
oa:     .=.+4       /область аргументов из двух слов: указатель
                    /   стека до вызова и адрес одного аргумента;
                    /
base:               /область под переменные;
a_      = base+2    /адрес a;
b_      = base+6    /адрес b;
        .=.+10      /размер области 10 байт;
.text
.globl main         /объявление внешнего имени.
```

В генерируемой ассемблерной программе внутренние имена переменных формируются из фортрановских с добавлением символа подчеркивания "_", а к именам подпрограмм добавляется символ точки ".".

Функция:

```
      real function c(a)
      real a
      c=a*a-2.*a+10.
      return
      end
```

Ассемблерный код:

```
c.: c_              /адрес области для возвращаемого значения;
                    / это первое слово любой подпрограммы;
        rval4p; a_  / значение аргумента a записать в стек;
        rval4p; a_  /       -"-
        rmp4        /вычисление a*a и запись результата в стек;
        rval4; c0   /запись константы 2. в стек;
        rval4p; a_  /запись аргумента в стек;
        rmp4        /умножение;
        rsb4        /вычитание;
                    /в стеке a*a-2.*a;
        rad4        /сложение, в стеке окончательный результат;
                    /
        gmv4; c_    /засылка результата в область возвращаемого
                    /      значения;
        retrn       /переход к подпрограмме возврата;
c0:     40400; 0    /константа 2.;
c1:     41040; 0    / константа 10.;
.bss
base:               /область переменных;
c_      =base+2     /результат;
a_      =2.         /смещение для адреса аргумента;
                    /
        .=.+6
.text
.globl c.
```


В приведенных примерах показана типичная процедура обращения к функции. Аналогично вызываются подпрограммы. Приведем тексты подпрограмм библиотеки интерпретации call и retrn.

Подпрограмма call:

```
call:   mov   (r4)+,r0      /адрес функции в r0;
        mov   r3,-(sp)      /сохраняем r3;
        mov   (r4)+,r3      /загружаем указатель на область
                            /       аргументов;
        mov   r4,-(sp)      /сохраняем r4, который теперь указыва-
                            /     ет на количество аргументов;
                            /
        mov   r0,r4         /адрес функции в r4;
        mov   (r4)+,-(sp)   /запись в стек адреса области для
                            /        результата;
        jmp   *(r4)+        /переход к интерпретации первой
                            /     команды функции;
```

Подпрограмма retrn:

```
retrn:  mov   r3,r0         /запись адреса oa в r0;
        mov   (sp)+,r1      /запись адреса   области для
                            /     результата в r1;
        mov   (sp)+,r4      /r4 указывает на количество аргументов;
                            /
        mov   (sp)+,r3      /r3 - адрес oa для вызывающей под-
                            /     программы;
        mov   (r0), sp      /восстановление исходного состояния
                            /     стека до вызова;
        tst   (r4)+         /r4 указывает на длину результата;
        mov   (r4)+,r0      /длина результата в r0;
        inc   r0            /округление длины до четного числа;
        bic   $1,r0         /
        add   r0,r1         /пересылка результата в стек;
1:      sub   $2,r0
        blt   2f
        mov   -(r1),-(sp)
        br    1b
2:      jmp   *(r4)+        /возврат к интерпретации команд
                            /   в вызывающей программе.
```

2.5. Рекомендации по созданию ассемблерных подпрограмм.

Ниже приведена общая структура подпрограммы на ассемблере, которая вызывается из фортрановской программы.

```
        .globl name.        /объявление внешнего имени;
name.:  result              /адрес результата функции;
        begin               /переход на следующую команду через
                            /     интерпретатор;
begin:  ...                 /выполнение необходимых действий;
        ...
```

```
        ...
        jmp retrn       /возврат из подпрограммы;
        .bss
result: .=.+4           /резервирование памяти под результат.
                        /
```

Количество переданных аргументов можно извлечь командой mov *2(sp),... Доступ к адресам аргументов осуществляется через регистр r3:

адрес первого аргумента - 2(r3)
адрес второго аргумента - 4(r3)
и т.д.

Вызов этой подпрограммы из фортрана:

i=name(...) или call name(...)

2.6. Связь с другими языками

Поскольку в других языках /например, Си [8]/ вызов подпрограмм оформляется иначе, то прямо их вызвать из фортрановских программ нельзя. В настоящее время это можно делать, редактируя текстовый промежуточный ассемблерный файл, заменяя имя call другим, например, callc - для вызова подпрограмм на языке Си. Необходимо также указать имя вызываемой подпрограммы.

Для решения обратной задачи рекомендуется использовать подпрограммы на языке ассемблер, вызываемые из соответствующего языка.

3. БИБЛИОТЕКИ

Выполнение фортрановских программ - это, по-существу, непрерывная последовательность вызовов библиотечных подпрограмм. Содержимое библиотеки filib.a предназначено для интерпретации "сшитого" кода; libf.a содержит встроенные функции фортрана, а также ряд дополнительных функций и подпрограмм.

Для выполнения арифметических операций типа real используются команды СМ-4: FADD, FSUB, FMUL и FDIV. Для увеличения скорости работы программ применяются улучшенные алгоритмы арифметических операций integer*4 /тип целых чисел по умолчанию/.Так, для умножения и деления на этапе загрузки операндов в регистры /т.к. аппаратные команды 16-битового умножения и деления работают с аргументами в регистрах/ выполняется оценка значений этих операндов, и далее работает соответствующая ветвь подпрограммы. С учетом того, что результат должен размещаться в 32 разрядах, исполнительные ветви алгоритма умножения содержат всего одно или два аппаратных умножения, а в случае переполнения они сразу выходят на ошибку. Алгоритм деления также основан на подобном подходе.

3.1. Библиотека интерпретации filib.a

В состав библиотеки входят 19 модулей. Общий объем библиотеки - 27 Кбайт.

3.1.1. Назначение модулей библиотеки

io - выполнение операций ввода-вывода и форматных преобразований.
r1 - пересылка из стека в стек, оператор цикла и все типы операторов go to.
r2 - сравнение целых чисел, логические операции.
r3 - логический и арифметический if.
r4 - арифметика integer*2, преобразования между целыми и логическими типами.
r5 - арифметика integer*4.
r6 - арифметика real*4, real*8, взаимные преобразования между целыми и вещественными типами.
r7 - работа с параметрами функций и подпрограмм.
r8, r9, ra - возведение в степень для целых и вещественных показателей и степеней.
rb - вызов и возврат из подпрограмм.
rc - комплексная арифметика.
rd - создание списка ввода-вывода для массива.
re - сравнение чисел real*8.
rf - сравнение чисел real*4.
rg - оператор присвоения.
rh - работа с массивами.
rx - пустой модуль.

В каждом модуле содержится несколько подпрограмм.Общее их количество в библиотеке - 184. При обращении к любой подпрограмме в состав исполняемой программы включается весь модуль.

3.1.2. Скоростные характеристики

Основная цель работы - повышение скорости выполнения программ - достигнута, в основном, за счет улучшения подпрограмм библиотеки интерпретации, и главный вклад здесь дают арифметические подпрограммы. Ниже приведены некоторые данные о скорости выполнения арифметических операций/оп/с/ и ряда других подпрограмм. Операции выполняются над операндами, помещенными в стек. Загрузка и выгрузка операнда из стека, сравнимые по времени со сложением integer*4, здесь не учитываются. В скобках указано полученное ускорение по сравнению с эмуляцией команд FPP.

	integer*2	integer*4	real*4	real*8
сложение -	150000/1/	61000/430/	36000/110/	4000/13/
умножение-	52000/1/	22000/140/	23000/95/	1500/6,5/
деление -	45000/1/	5500/33/	17000/70/	600/2,5/

вызов процедуры с двумя параметрами - 12000/1/
преобразование integer*2 в real*4 - 22000/55/
преобразование real*4 в integer*2 - 19000/60/
цикл do, одна итерация - 16000/1/

3.2. Библиотека libf.a

Эта библиотека содержит 84 функции и подпрограммы. Большая часть их реализует встроенные функции фортрана. Размер библиотеки - около 27 Кбайт.

3.2.1. Скоростные характеристики

Модификация библиотеки libf.a позволила значительно повысить скорость выполнения встроенных функций. Ниже приводятся оценки вычисления наиболее сложных математических функций /число функций в секунду/. В скобках - полученное ускорение.

r=alog(r)	1700(75)	d=dlog(d)	85(3.5)
r=atan(r)	1450(85)	d=datan(d)	75(4.5)
r=cos(r)	1500(90)	d=dcos(d)	70(4.0)
r=exp(r)	1900(80)	d=dexp(d)	110(5.0)
r=sin(r)	1700(100)	d=dsin(d)	75(4.5)
r=sqrt(r)	3000(100)	d=dsqrt(d)	125(4.5)

3.2.2. Дополнительные функции и подпрограммы

Кроме стандартных встроенных функций в библиотеке содержится ряд дополнительных функций и подпрограмм.

Это, во-первых, функции двойной комплексной арифметики: dcabs, dccos, dcexp, dclog, dcsin, dcsqrt, dimag, dconjg.

Во-вторых, функции для работы с файловой системой ОС UNIX: iopen, icreat, iclose, iread, iwite, ilseek, ierrno.

Остальные функции предназначены для:

srand и rand - генерации случайных чисел;
ctime и itime - запроса времени дня;
setfil - связывания канала ввода-вывода с определенным файлом;
getarg и iargc - работы с аргументами программы;
ierror - обработки ошибок при выполнении программы;
onkill - однократного перехвата прерывания <control>-c;
unicom - выполнения команды ОС UNIX;
nice - изменения приоритета программы.

4. ОЦЕНКА СКОРОСТИ ВЫПОЛНЕНИЯ ФОРТРАНОВСКИХ ПРОГРАММ ПО ТЕСТУ WHETSTONE BENCHMARK

Этот достаточно широко распространенный тест позволяет оценить скорость выполнения фортрановских программ. В табл.2 приведены данные для нескольких ЭВМ с различными операционными системами. В результате модификации библиотек фортрана-4 скорость выполнения этого теста на ЭВМ СМ-4 возросла примерно в 40 раз.

Таблица 2

ЭВМ, операционная система, особенности компилятора	Скорость выполнения фортрановских программ, в тыс. плавающих операций/с
PDP-11/40 UNIX, эмуляция	1,9
PDP-11/40 UNIX, без эмуляции	75
PDP-11/40 RT-11, inline code	101
PDP-11/40 RT-11, threaded code	76,5
PDP-11/45 UNIX	105
PDP-11/45 RT-11, inline code	170
PDP-11/45 RT-11, threaded code	150
VAX-11/780, VMS	1600

Авторы выражают благодарность Б.А.Морозову, А.Г.Кареву и О.А.Бессонову за многочисленные полезные советы и помощь в работе.

ЛИТЕРАТУРА

1. Ritchie D.M., Thompson K. Bell System Technical Journal, 1978, vol.57, No.6, p.1905.
2. Инструментальная мобильная операционная система - ИНМОС, "Финансы и статистика", М., 1985.
3. Ritchie D.M., "UNIX Assembler Reference Manual". Documents for the PWB/UNIX Time Sharing System, Bell Laboratories, 1977, 18.
4. "PDP-11 FORTRAN Language Reference Manual", (DEC-11-LFLRA-B-D), Maynard, Massachusetts, 1974.
5. Грундт Ф. "Программирование на языке фортран-4", "Мир", М., 1976.
6. Катцан Г. "Язык фортран-77", "Мир", М., 1982.
7. Меткалф М. "Оптимизация в фортране", "Мир", М., 1985.
8. Керниган Б.,Ритчи Д.,Фьюэр А. Язык программирования Си. Задачи по языку Си."Финансы и статистика", М., 1985.

Рукопись поступила в издательский отдел
25 марта 1986 года.

Вниманию организаций и лиц, заинтересованных в получении публикаций Объединенного института ядерных исследований

Принимается подписка на препринты и сообщения Объединенного института ядерных исследований.

Установлена следующая стоимость подписки на 12 месяцев на издания ОИЯИ, включая пересылку, по отдельным тематическим категориям:

ИНДЕКС	ТЕМАТИКА	Цена подписки на год
1.	Экспериментальная физика высоких энергий	10 р. 80 коп.
2.	Теоретическая физика высоких энергий	17 р. 80 коп.
3.	Экспериментальная нейтронная физика	4 р. 80 коп.
4.	Теоретическая физика низких энергий	8 р. 80 коп.
5.	Математика	4 р. 80 коп.
6.	Ядерная спектроскопия и радиохимия	4 р. 80 коп.
7.	Физика тяжелых ионов	2 р. 85 коп.
8.	Криогеника	2 р. 85 коп.
9.	Ускорители	7 р. 80 коп.
10.	Автоматизация обработки экспериментальных данных	7 р. 80 коп.
11.	Вычислительная математика и техника	6 р. 80 коп.
12.	Химия	1 р. 70 коп.
13.	Техника физического эксперимента	8 р. 80 коп.
14.	Исследования твердых тел и жидкостей ядерными методами	1 р. 70 коп.
15.	Экспериментальная физика ядерных реакций при низких энергиях	1 р. 50 коп.
16.	Дозиметрия и физика защиты	1 р. 90 коп.
17.	Теория конденсированного состояния	6 р. 80 коп.
18.	Использование результатов и методов фундаментальных физических исследований в смежных областях науки и техники	2 р. 35 коп.
19.	Биофизика	1 р. 20 коп.

Подписка может быть оформлена с любого месяца текущего года.

По всем вопросам оформления подписки следует обращаться в издательский отдел ОИЯИ по адресу: 101000 Москва, Главпочтамт, п/я 79.

Вицев В.В., Егошин Л.А., Ломов А.К. Р11-86-172
Компилятор и библиотеки языка фортран-4 в операционной системе UNIX для ЭВМ СМ-4

Описывается модифицированный компилятор языка фортран-4, позволяющий расширить сферу использования систем разделения времени типа ОС UNIX на ЭВМ СМ-4. Его скоростные характеристики не уступают характеристикам компилятора фортрана в ОС РАФОС. Наличие препроцессора позволяет легко адаптировать программы, написанные на различных версиях языка фортран-4.

Работа выполнена в Серпуховском научно-экспериментальном отделе ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1986

Перевод О.С.Виноградовой

Vitsev V.V., Egoshin L.A., Lomov A.K. P11-86-172
Fortran-4 Compiler and Libraries in the UNIX Operating System for SM-4 Computer

A modified compiler of the Fortran-4 language is described. It permits to expand the field of using on the SM-4 computer time sharing systems of OS UNIX type. Execution speed is as good as that of Fortran compiler in OS RAFOS. The presence of the preprocessor permits to easily adopt the programs written in different versions of the Fortran-4 language.

The investigation has been performed at the Serpukhov Scientific-Experimental Department, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1986