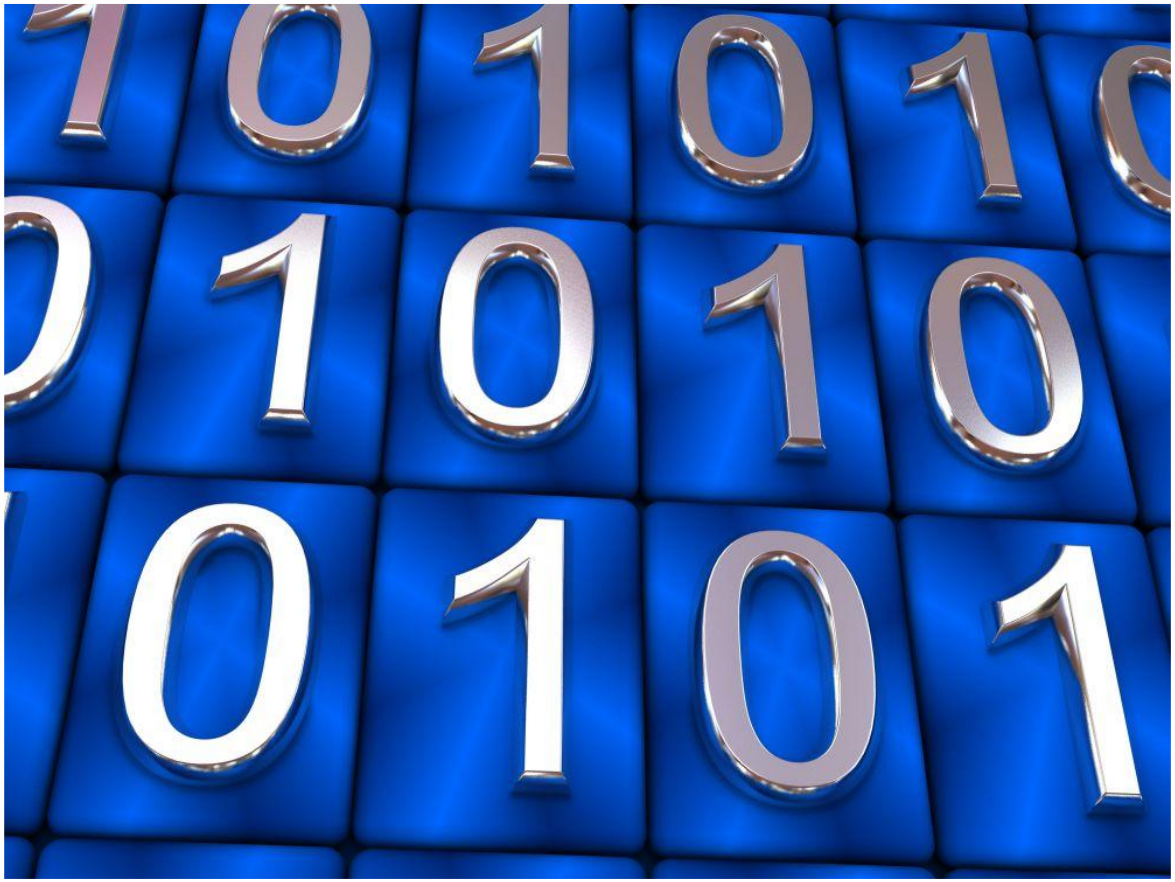
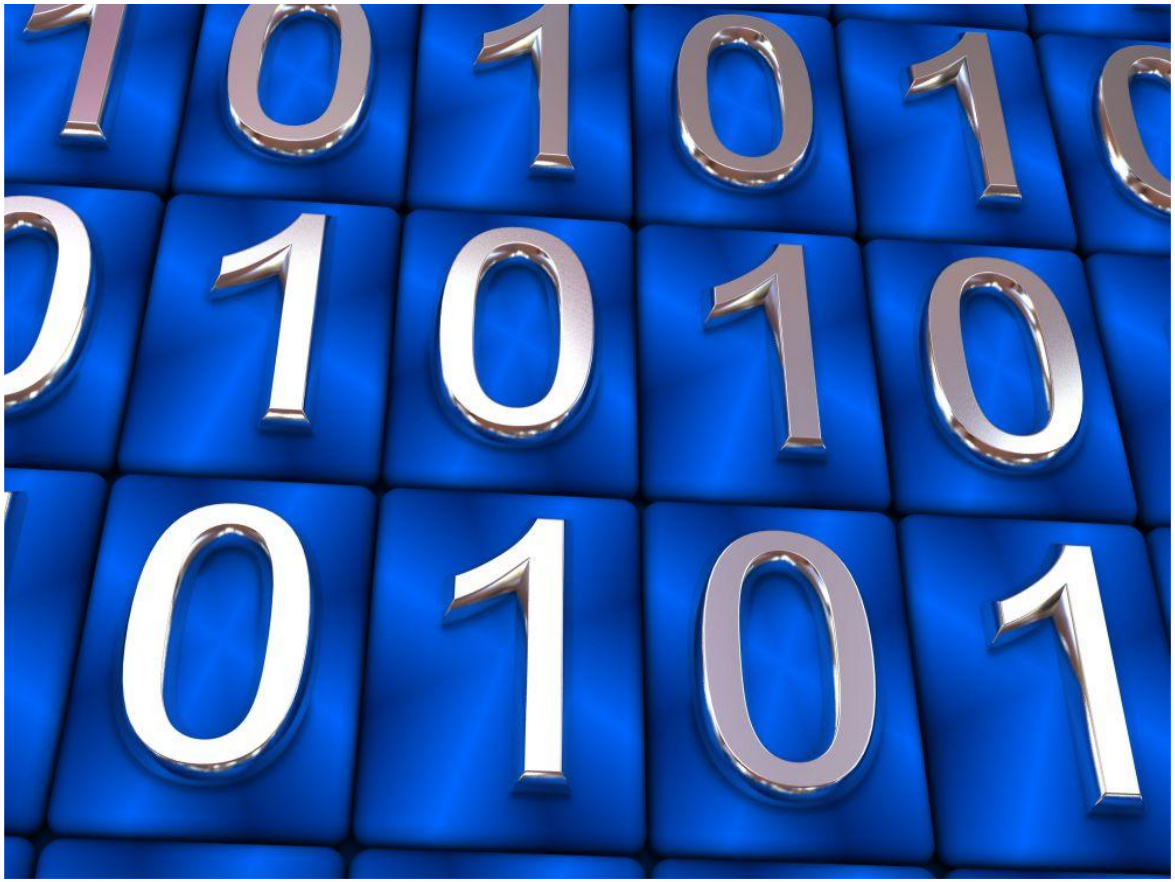
ЯЗЫКИ ПРОГРАММИРОВАНИЯ

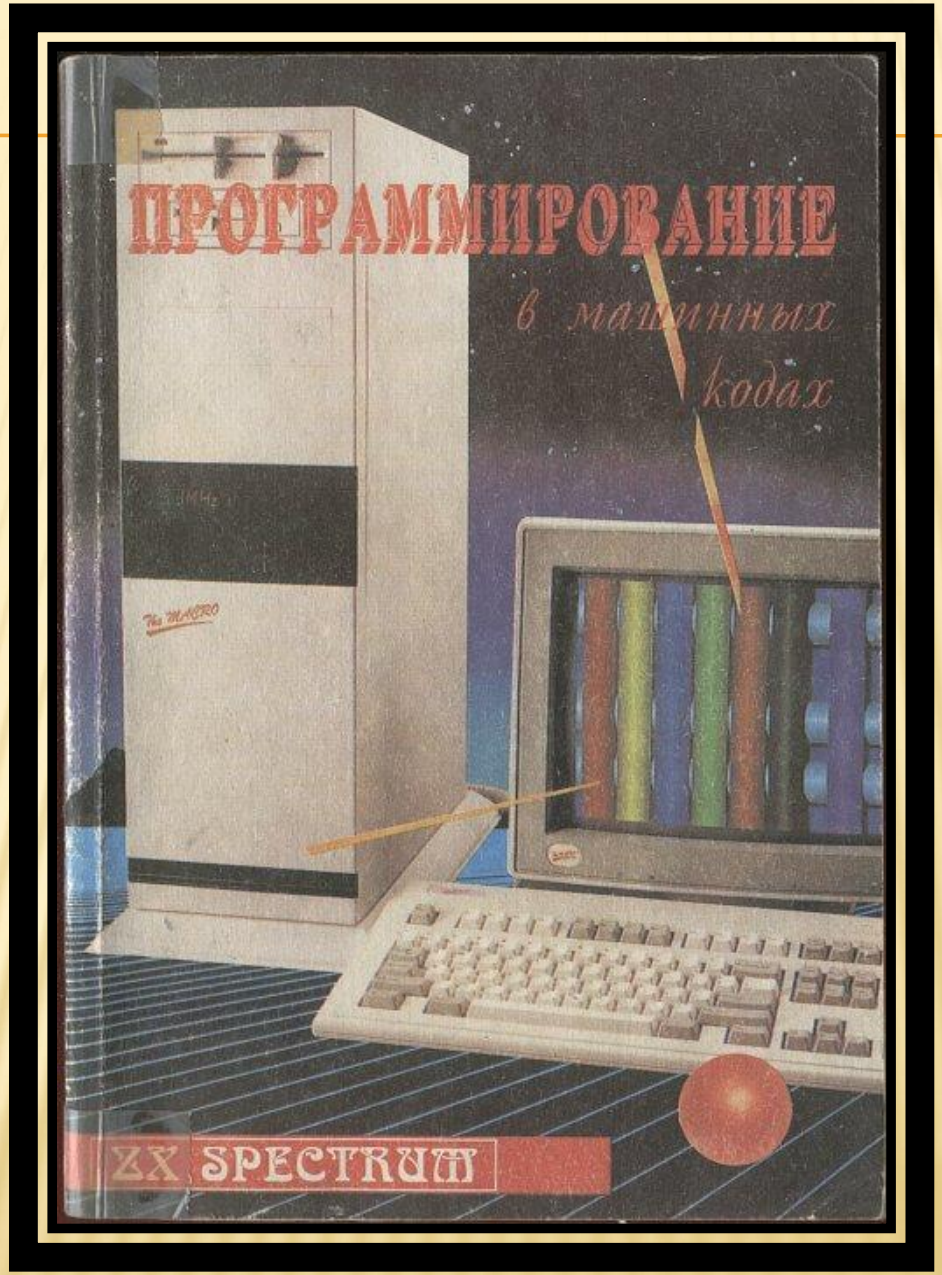


- Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования определяет набор лекс
зада



ПЕРВЫЕ УНИВЕРСАЛЬНЫЕ ЯЗЫКИ

Первые программы писались на машинном языке. Программисты обязаны были знать архитектуру машины досконально. Программы были достаточно простыми, что обуславливалось, во-первых, весьма ограниченными возможностями этих машин, и, во-вторых, большой сложностью разработки и, главное, отладки программ непосредственно на машинном языке. Вместе с тем такой способ разработки давал программисту просто невероятную власть над системой. Становилось возможным использование хитроумных алгоритмов и способов организации программ. Например, могла применяться такая возможность, как самомодифицирующийся код. Знание двоичного представления команд позволяло иногда не хранить некоторые данные отдельно, а встраивать их в код как команды.


ПРОГРАММИРОВАНИЕ
в машинных
кодах
ZX SPECTRUM

СИСТЕМА КОМАНД ЭВМ-220

Операции над числами (p, q, r – порядки чисел x, y, z по A_1', A_2', A_3')

Операция	КОП ОиН	КОП БО	КОП БН	КОП БОиБН	ω=1	t мксек
Сложение	01	21	41	61		
Вычитание	02	22	42	62	Z<0	~28,5
Вычитание модулей	03	23	43	63		
Деление	04	24	—	—		~107
Умножение	05	25	45	65	$Y>100_8$	51
Извлечение кв. корня	44	64	—	—		272
Вывод младших разрядов произведения	47					24

Операции изменения порядка

Операция	КОП	Результат	ω=1	t мксек
Сложение порядка с A_1'	06	$r=q+(A_1'^{(1)}-100)$		
Сложение порядка с P	26	$r=q+(p-100)$	$r>100_8$	24
Вычитание порядка из A_1'	46	$r=q-(A_1'^{(1)}-100)$		
Вычитание порядка из P	66	$r=q-(p-100)$		

Команда останова

КОП	Совмещенные операции
77	Вызов $[A_1']$ и $[A_2']$ на P_1 и P_2 гашение $[A_3']$ при продолжении

Операции изменения [РА]

КОП	Код в РА	Код в A_3'	t мксек
52	A_2'	0520000 A_1' 0000	24
72	зона $A_2[A_2']$	0520000 A_1' 0000	

Команды ввода

Операция	КОП	A_1'	A_2'	A_3'
Ввод с остановом при несовпадении КΣ	10	Aн МОЗУ	КРА	АКΣ
	от кн. ввод	0001	0000	0000
Ввод без ост. при несовпад КΣ	30	Aн МОЗУ	КРА	АКΣ

Операция изменения регистров приращения

КОП	A_1'	A_2'	код заносимый в A_3'	t мксек
57	2,1р – РПМЛ 5,4р – РПМЕ 7р – Пр А3 8р – Пр А2 9р – Пр А1 10р – Пр КРА 11р – Пр МЛ 12р – Пр МБ	3,2,1р – РПА3 6,5,4р – РПА2 9,8,7р – РПА1 12,11,10р – РПКРА	05736÷36р, 30÷25, 24÷13, [illegible] признак 12–7р A_1; РПМОЗУ РАМОЗУ старые значения	24

Логические операции (ξ, η, ζ – значения кодов по A_1', A_2', A_3')

Операция	КОП	Результат	ω=1	t мксек
Сравнение	15	$\zeta_k=\lvert\xi_k-\eta_k\rvert$	ζ=0	24
Сравнение с ост	35	аналогично 15, но ОСТ при $\zeta\neq0$		
Логическое умножение	55	$\zeta_k=\xi_k\wedge\eta_k$		
Логическое сложение	75	$\zeta_k=\xi_k\vee\eta_k$		

Специальные операции над кодами

Операция	КОП	Результат	ω=1	t мксек
Цикл сложения	07	$\zeta'=(\xi'+\eta')\bmod(2^{12}-1)$ $\zeta''=(\xi''+\eta'')\bmod(2^{36}-1)$	$\xi''\geq2^{36}$	24
Цикл вычитания	27	$\zeta'=(\xi'-\eta')\bmod(2^{12}-1)$ $\zeta''=(\xi''-\eta'')\bmod(2^{36}-1)$	$\xi''<\eta''$	
Сложение мантисс	13	$\zeta'=\xi'$ $\zeta''=(\xi''+\eta'')\bmod 2^{36}$	$\xi''\geq2^{36}$	
Вычитание мантисс	33	$\zeta'=\xi'$ $\zeta''=(\xi''-\eta'')\bmod 2^{36}$	$\xi''<\eta''$	
Сложение КОПов	53	$\zeta'=(\xi'+\eta')\bmod 2^{9}$	$\xi'\geq2^{9}$	
Вычитание КОПов	73	$\zeta'=(\xi'-\eta')\bmod 2^{9}$ $\zeta''=\xi''$	$\xi'<\eta'$	
Цикл сдвиг	67	$\zeta_i=\xi_{(i+24)\bmod 2^{48}}$ A_2' безразличен	—	

Операции в М2, выполняемые по сигналам из М1

Операция		Информация переданная на РБФ М2 из М1
Выборка команды в М2 по адресу из М1	$АР2_{м2}$	40÷38р РП адреса в М2; 34р без контр; 24÷13р адрес команды
Обмен кодами М1 ⇄ М2	$АР1_{м2}$	40÷38р РПА2М2; 36÷25р УЧ; 24÷13р Aн МОЗУ

Операции выполняемые по сигналам от линии связи и аналоговой машины

Операция		Результат
Авторазрыв от ЛС	АРЛС	Выборка команды из яч-ки 0022 „0" МОЗУ без занесения 0022 на КРА
Авторазрыв от АМ	АРАМ	Выборка команды из яч-ки 0023 „0" МОЗУ без занесения 0023 на КРА

Операции сдвига

Операция	КОП	Величина сдвига	ω=1	t мксек
Сдвиг кода по A_1'	54	$\zeta_{k+s}=\eta_k$; $S=A_1'^{(2)}-100$; A_1' [illegible] A_1'	ζ=0	24+1,5S
Сдвиг кода по P	74	$\zeta_{k+s}=\eta_k$; $S=p-100$	ζ=0	24+1,5S
Сдвиг мантиссы по A_1'	14	$\zeta''_{k+s}=\eta''_k$; $S=A_1'^{(1)}-100$	ζ=0	24+1,5S
Сдвиг мантиссы по P	34	$\zeta''_{k+s}=\eta''_k$; $S=p-100$	ζ=0	24+1,5S

Операции засылки

Операция	КОП	Результат	ω=1	t мксек
Выборка из динамики	17	Засылка кода $[Z]\to Y$, Y-ячейка по A_3'	ζ=0	≈90
Засылка в динамику	37	Засылка кода $[Y]\to Z$, Y-ячейка по A_3'		
Засылка кода	00	$[A_1']\to A_3'$, A_2' безразличен	—	24
Засылка кода с КЗУ	20	$[КЗУ]\to A_3'$, 3 мл. разряда A_1' – Nк ЗУ	—	
Анализ окончания раб АМ	20	A_1'–0000 → переход A_2' при „0" Траб АМ	—	

Команды передачи управления с изменением [РА]

КОП	Код в РА	Передача управл. в A_2'	Передача управлен. в КРА+1	t мксек
11	A_3'	РА < A_1' и ω=1	РА ≥ A_1' или ω=0	24
31		РА ≥ A_1' и ω=1	РА < A_1' или ω=0	
51		РА < A_1' и ω=0	РА ≥ A_1' или ω=1	
71		РА ≥ A_1' и ω=0	РА < A_1' или ω=1	
12		РА < A_1'	РА ≥ A_1'	
32		РА ≥ A_1'	РА < A_1'	
40		РА < зона $A_2[A_1']$	РА ≥ зона $A_2[A_1']$	
60		РА ≥ зона $A_2[A_1']$	РА < зона $A_2[A_1']$	

Команды передачи управления с занесением в A_3'

КОП	Код, заносимый в A_3'	Передача управления в A_2'	Передача управления в КРА+1	t мксек
16	0160000 A_1' 0000	Всегда	—	24
56	$[A_1']$	Всегда	—	
36		ω=1	ω=0	
76		ω=0	ω=1	

Операции, выполняемые по командам Ма и Мб

Ма (КОП 50), Мб (КОП 70)

Режим		УЧ	A_1' 12 (БЛМ)	11 (ωК)	10 (МЧ/БЧ)	9 (БЛ Ост)	8 (Пф)	7 (Пч)	6 (РЛ)	5 (МЛ)	4 (МБ)	3 (За/Чт)	2 (№ МЗУ)	1 (№ МЗУ)	Ма A_2'	Ма A_3'	t мксек	Мб A_1'	Мб A_2'	Мб A_3'
Обмен с МБ		0010			x		0	0	0	0	1	1/0			10÷1р номер числа в тракте МБ 12÷11р номер тракта МБ 10÷1р номер зоны МЛ	Aк МОЗУ	24	Aн МОЗУ	КРА при несовп КΣ	АКΣ
Обмен с МЛ		0020			0/1		0	0	0	1	0	1/0								
Печать „8"		0500			x	1	0	1	0	0	0	x	x	x						
Печать „10"		0100			x	0	0	1	0	0	0	x	x	x						
Печать Ал.Ц		0140			x	0	0	1	1	0	0	x	x	x						
Перфорация		0200			x	x	1	0	0	0	0	x	x	x						
Накопление		0300		x	x	x	1	1	x	0	0	x	x	x						
Разметка МЛ		0040			0/1	x	0	0	1	0	0	x								
Подвод МЛ		0060	x	x	0/1	x	0	0	1	1	0	x								
Засылка состояния	$АР3_з$	0030	РПА2			РПА3			x	1	1	0	0	0	КРА	А Зп состояния	32,25	Заполняются только по команде Ма		
Возврат состояния	$АР3_в$	0034	РПА2			0	0	0	x	1	1	1	0	0	АЧт состояния	0000	38,25			
Выборка команды в М2 по адресу из М1	АР2	0033	РП адр ком-ды в М2			x	x	x	1	1	1	x	1	1	Адрес команды М2	безразличен				
Обмен кодами М1 ⇄ М2	АР1	0031			1	РПА2 М2				1	1	1/0	0	1	Aн МОЗУ М2	Aк МОЗУ		Aн МОЗУ	КРА	АКΣ
Работа с АМ	АМ	0032		x	x	x	x	1 ост	1	1	1	1/0	1	0	5÷1р нач. канал				Безразличны	

Примечание: Знак „X" означает, что состояние разряда безразлично.
Отсутствие знак[а оз]начает, что в разряде может быть „0" или „1"; A' – адрес исполнительный; [A] – содержимое ячейки с адресом A

1. Емкость промежуточного накопителя вывода ПНВ 1024 слова
2. После вывода содержимое ПНВ не стирается
3. При обмене с МБ возможен переход с данного барабана на следующий (4096→4097)
4. Команда останова имеет код только 77
5. Команда 20 при A_1'=0000 используется как команда перехода по признаку работы с аналоговой машиной
6. В НМЛ N3 = 0 используется только в режиме „Разметка" для запирания дефектных зон МЛ.

***

1. При чтении адреса с ПК „1" 1р РБФ – блокировка АВ Ост при несовпадении КΣ; „1" 13р РБФ – блокировка записи в МОЗУ последующего массива; 37÷38р РБФ(РПА1) → 13÷14р СмА
2. При A_1'=0 блокируется запись в МОЗУ последующего массива

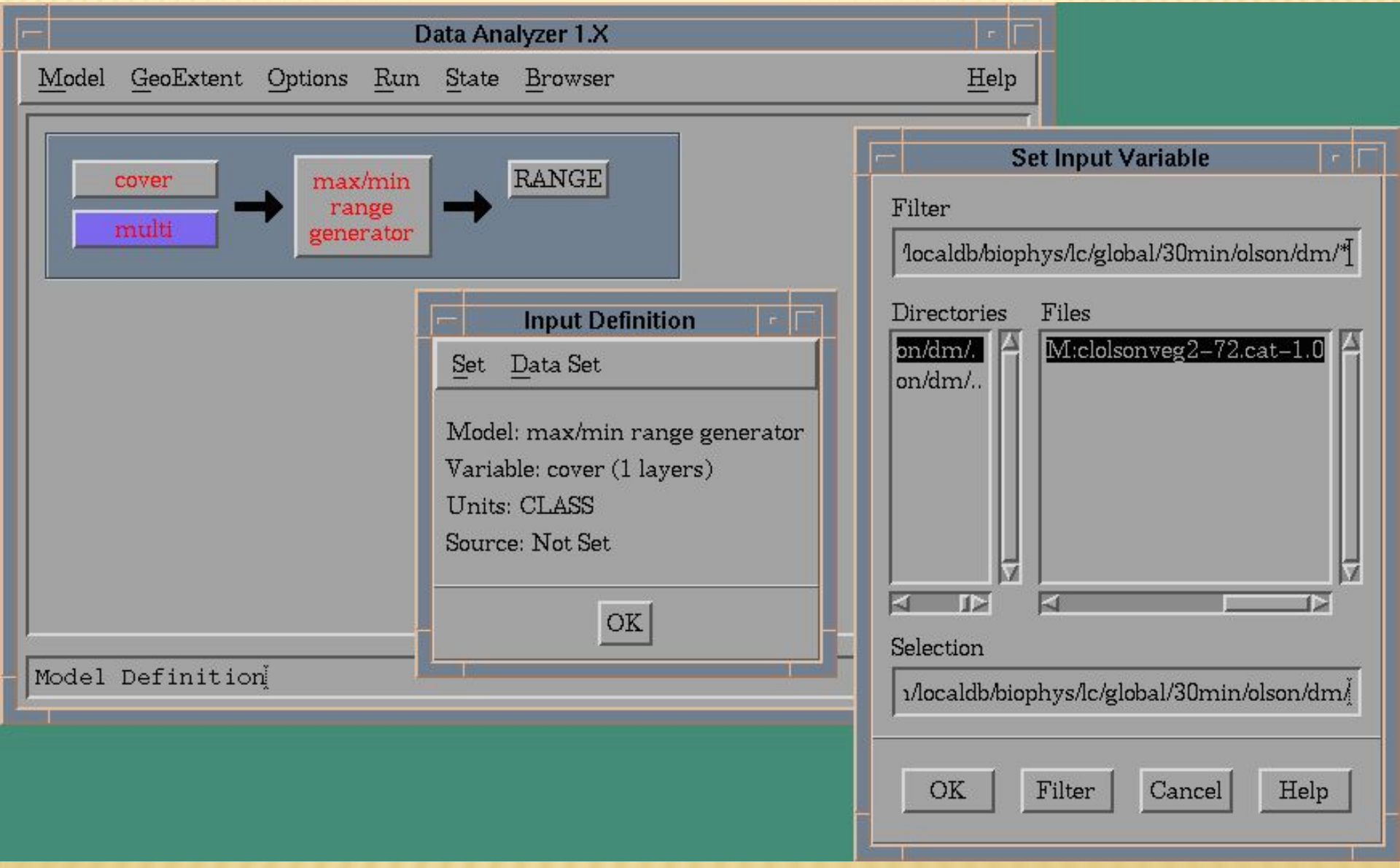
**
Обращение к МОЗУ по команде НРП производятся по старым значениям ИРП МОЗУ. РП изменяется только по командам ИРП, АР3в

*
Операции $АР1_{м2}$, $АР2_{м2}$, АРЛС, АРАМ не изменяют состояния машины.

Первым значительным шагом представляется переход к языку ассемблера. Программисту не надо было больше вникать в способы кодирования команд на аппаратном уровне. Появилась также возможность использования макросов и меток, что также упрощало создание, модификацию и отладку программ.

АССЕМБЛЕР

Вместе с тем, переход к новому языку таил в себе и некоторые отрицательные стороны. Возможности программистов сильно сократились. Кроме того, здесь впервые в истории развития программирования появились два представления программы: в исходных текстах и в откомпилированном виде. К концу ассемблерной эры возможность автоматической трансляции в обе стороны была утеряна. В связи с этим было разработано большое количество специальных программ-дизассемблеров, осуществляющих обратное преобразования, однако в большинстве случаев они с трудом могут разделить код и данные.


Data Analyzer 1.X
Model GeoExtent Options Run State Browser Help
cover
multi
max/min range generator
RANGE
Input Definition
Set Data Set
Model: max/min range generator
Variable: cover (1 layers)
Units: CLASS
Source: Not Set
OK
Model Definition
Set Input Variable
Filter
/localdb/biophys/lc/global/30min/olson/dm/*
Directories
Files
on/dm/.
on/dm/..
M:clolsonveg2-72.cat-1.0
Selection
ı/localdb/biophys/lc/global/30min/olson/dm/
OK
Filter
Cancel
Help

ФОРТРАН

Следующий шаг был сделан в 1954 году, когда был создан первый язык высокого уровня — Фортран. Впервые программист мог по-настоящему абстрагироваться от особенностей машинной архитектуры. Синтаксическая структура языка была достаточно сложна для машинной обработки в первую очередь из-за того, что пробелы как синтаксические единицы вообще не использовались. Это порождало массу возможностей для скрытых ошибок, таких, например:

В Фортране конструкция : “DO 10 I=1,100” описывает«цикл выполнения оператора при изменении индекса от 1 до 100» Если же здесь заменить запятую на точку, то получится оператор присваивания: DO10I = 1.100.

ФОРТРАН

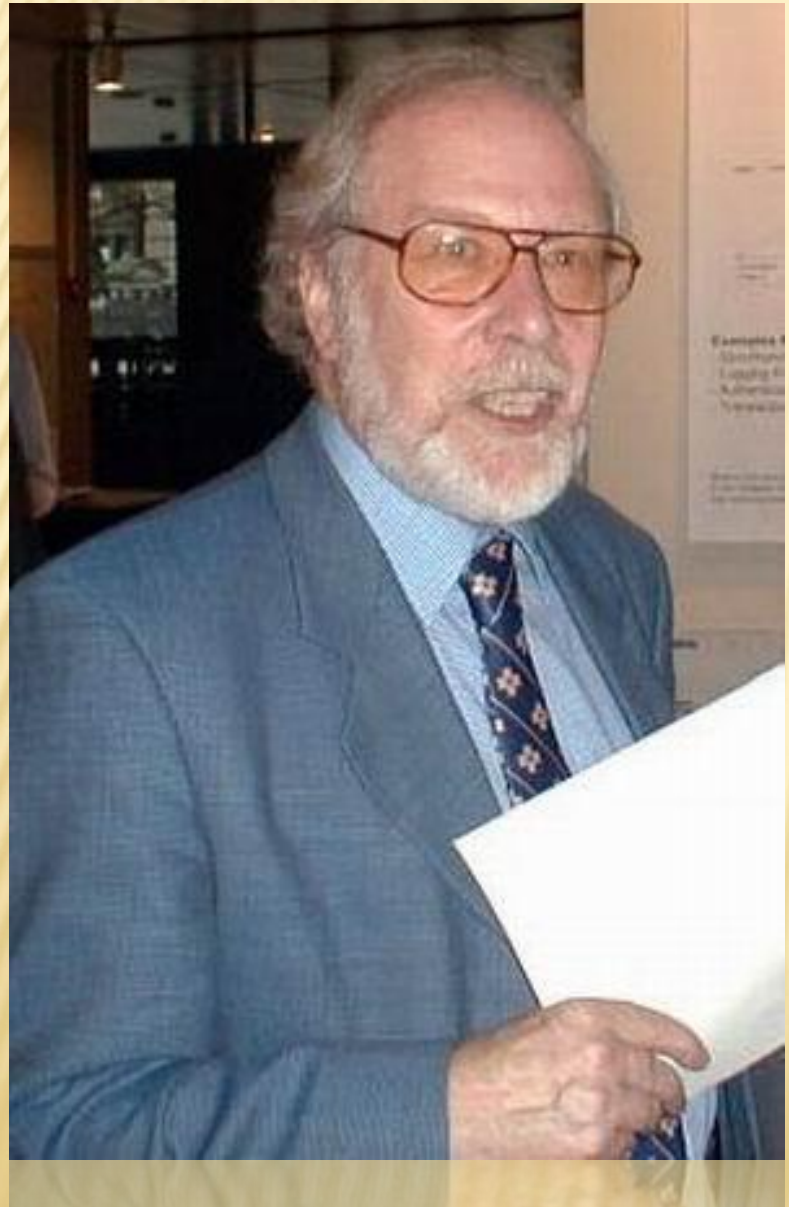
Язык Фортран использовался для научных вычислений. Он страдает от отсутствия многих привычных языковых конструкций и атрибутов, компилятор практически никак не проверяет синтаксически правильную программу с точки зрения корректности. По признанию самого Бэкуса, перед ними стояла задача скорее разработки компилятора, чем языка. Понимание самостоятельного значения языков программирования пришло позже.

ФОРТРАН

Появление Фортрана было встречено еще более яростной критикой, чем внедрение ассемблера. Через некоторое время пришло понимание того, что реализация больших проектов невозможна без применения языков высокого уровня. Мощность вычислительных машин росла, и с тем падением эффективности, которое раньше считалось угрожающим, стало возможным смириться. Преимущества же языков высокого уровня стали настолько очевидными, что побудили разработчиков к созданию новых языков, все более и более совершенных.

- 1960 г. – создание языка Cobol
- 1960 г. Петер Наур создал язык программирования Algol.
- 1963 г. – создание языка BASIC
- 1964 г. – корпорация IBM создала язык PL/1
- 1968 г. – новая версия языка Algol.

PASCAL-ПОДОБНЫЕ ЯЗЫКИ



В 1970 году Никлаусом Виртом был создал язык программирования Pascal. Язык замечателен тем, что это первый широко распространенный язык для структурного программирования. В этом языке также внедрена строгая проверка типов, что позволило выявлять многие ошибки на этапе компиляции.

СИ-ПОДОБНЫЕ ЯЗЫКИ

- В 1972 году Керниганом и Ритчи был создан язык программирования Си. Через 14 лет Бьярн Страуструп создал первую версию языка C++, добавив в язык C объектно-ориентированные черты. Язык стал основой для разработки современных больших и сложных проектов. В 1999–2000 годах в корпорации Microsoft был создан язык C#. Он в достаточной степени схож с Java (и задумывался как альтернатива последнему), но имеет и отличительные особенности. Ориентирован, в основном, на разработку многокомпонентных Интернет-приложений.

- В 1969 году был создан язык SETL — язык для описания операций над множествами. Основной структурой данных в языке является множество, а операции аналогичны математическим операциям над множествами.

- Perl – язык создавался в помощь системному администратору операционной системы Unix для обработки различного рода текстов и выделения нужной информации. Развился до мощного средства работы с текстами.

- Python – интерпретируемый, объектно-ориентированный язык программирования. По структуре и области применения близок к Perl, однако менее распространен и более строг и логичен.

Спасибо за внимание!)