Государственное бюджетное общеобразовательное учреждение
города Москвы
«Лицей «Вторая школа» имени В.Ф. Овчинникова»

Проектная работа
«Эмулятор троичного компьютера»

Выполнил: ученик 8 Г класса Волынец Павел Романович

Руководитель проекта: Маркелова Наталья Романовна

Консультант: Волынец Роман Владимирович

Москва, 2023

Содержание

Термины

Эмулятор — программа, которая позволяет одной цифровой системе воссоздать свойства другой внутри себя.

Троичный компьютер — компьютер, основанный на троичной системе счисления.

Трит — минимальная единица информации в троичном компьютере (по аналогии с битом)

Машинное слово — фрагмент данных фиксированного размера, обрабатываемый как единое целое с помощью набора команд или аппаратного обеспечения процессора.

Регистр — устройство для записи, хранения и считывания данных и выполнения различных операций над ними

Введение

В настоящее время все компьютеры – двоичные, и подавляющее большинство людей даже не задумывается над тем, что компьютеры могут быть основаны не на двоичной системе счисления. Но если над этим задуматься, то становится далеко не так очевидно, что компьютеры должны быть двоичными. Даже, наоборот, оказывается, что у некоторых других систем счисления (например, у троичной) есть свои преимущества.

В нашей работе мы попытались рассказать о плюсах и минусах систем счисления для компьютеров и о выборе лучшей из них.

Вопрос выбора наиболее эффективной системы счисления для компьютеров до сих пор был не таким важным (поскольку повсеместно используемая технология производства двоичных микросхем и двоичных процессоров долгое время обеспечивала необходимый рост производительности компьютеров за счет миниатюризации элементов). Но уже в 2003-2005 гг. стало заметно, что известный Закон Мура [1], согласно которому количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается каждые 2 года, перестает действовать [2] (рис.1). Поэтому производители процессоров стали использовать различные другие технологии повышения производительности процессоров, которые тем не менее, не помогли обеспечить такой рост производительности, который должен был быть в соответствии с Законом Мура.

Переход на другую (более эффективную) систему счисления может быть как раз актуален в связи с тем, что возможности миниатюризация современной элементной базы подходят к своему пределу.

В этом проекте мы рассматриваем компьютеры, основанные на троичной системе счисления, и говорим о её преимуществах и недостатках.

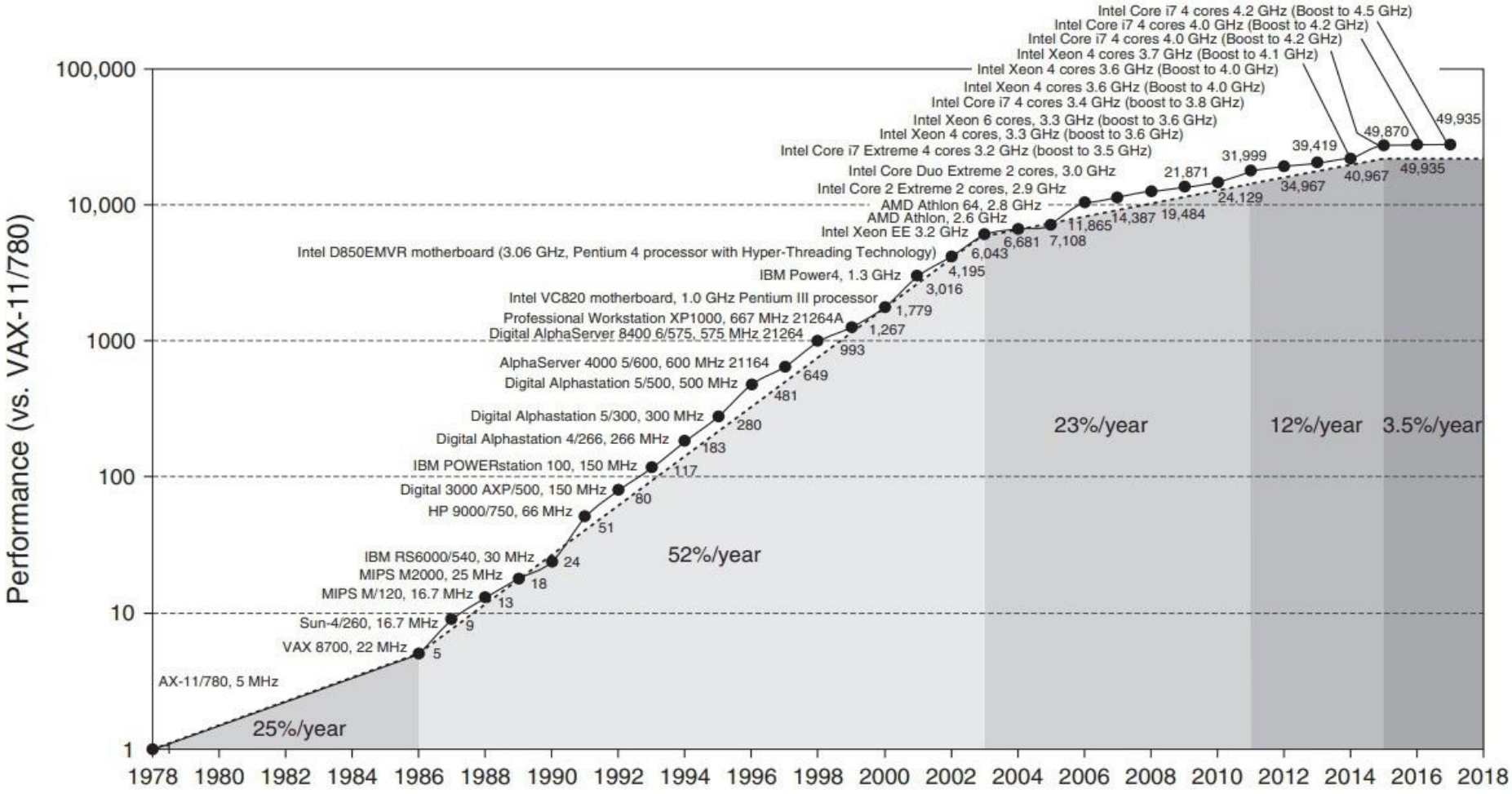

Рис. 1. Конец действия Закона Мура.

В подтверждение актуальности темы изучения и использования троичных компьютеров можно также привести такой факт: для студентов 4 курса читается спецкурс «Введение в троичную информатику» [3].

Цель работы – обоснование выбора троичной системы счисления, изучение особенностей реализации троичных компьютеров и написание эмулятора троичного компьютера (на базе двоичного компьютера).

Задачи:

1) Изучить особенности разных систем счисления.
2) Изучить литературу по устройству троичного и двоичного компьютера.
3) Спроектировать архитектуру троичного компьютера.
4) Написать код эмулятора троичного компьютера.
5) Придумать удобный способ ввода информации в эмулятор.
6) Написать тестовую программу для эмулятора.

1. Поиск и исследование информации

1.1 История

Интерес к троичной системе счисления появился довольно давно. В 1203 году Фибоначчи исследовал «задачу о гирях» [4]. В задаче требуется найти наименьшее число гирь, которое требуется для взвешивания на рычажных весах любого веса, меньше заданного. У этой задачи есть два варианта формулировки и решения: гири могут лежать только на одной из чаш или на обеих. В первом случае ответом будут гири с весами 1, 2, 4, 8, …, а во втором - 1, 3, 9, 27, …. Это объясняется тем, что во втором случае можно использовать не только сумму весов гирь, но и их разность.

В первом случае используется гири с весами, которые являются степенями 2, т.е. это двоичная система счисления, а во втором – гири с весами, которые являются степенями 3, т.е. это троичная система счисления. Из-за

того, что гири можно класть на любую чашу весов, они учитываются либо со знаком «плюс», либо со знаком «минус». Такая троичная система счисления называется *сбалансированной* или *симметричной*. В этой системе используются не цифры 0, 1 и 2, а цифры -1, 0 и 1 (или символы -, 0, +). Она симметрична относительно 0, из чего следуют несколько свойств этой системы, о которых будет рассказано ниже.

В 1840 году Томас Фаулер построил механический троичный умножитель с регистром результата [5], который содержал 55 троичных значений (элементов) в позиционной записи числа.

В 1958 году Николай Павлович Брусенцов построил первую электронную троичную ЭВМ «Сетунь» [6], которая начиная с 1959 года стала выпускаться серийно на Казанском заводе математических машин.

В 1970 Н. П. Брусенцов разработал вторую электронную троичную ЭВМ «Сетунь-70» [7], но эта разработка, к сожалению, в производство не пошла.

ЭВМ «Сетунь» – первая и единственная в мире троичная ЭВМ, которая выпускалась серийно. В ней триты представлялись с помощью двух битов (10 = +, 00 = 0, 01 = -, сочетание «11» не использовалось), реализованных с помощью магнитных усилителей на ферритовых сердечниках. Машинное слово было равно 18 тритам, или 6 трайтам (трайт – это 6 тритов).

Более подробную информацию об ЭВМ “Сетунь» можно найти здесь [6].

Всего было выпущено 49 ЭВМ «Сетунь» образца 1958 года. До сегодняшнего дня ни одна из них не сохранилась. В фондохранилище Политехнического музея хранятся только пульт от машины 1958 года и конструкторский образец машины «Сетунь-70» (см. Приложение).

1.2 Преимущества троичной систем счисления

Сначала немного скажем про другие системы счисления.

Чем больше основание системы счисления, тем сложнее устройство элемента, хранящего значение, и тем сложнее таблица операций (таблица сложения, таблица умножения), что ведёт к усложнению архитектуры процессора и замедлению его работы. Поэтому для компьютера подходят только системы с небольшим основанием. То есть двоичная, троичная, четверичная, пятеричная. Двоичная и четверичная системы - на самом деле одно и то же, ведь два бита соответствуют одному разряду в четверичной, но при этом четверичная система счисления гораздо сложнее.

Пятеричная система, даже симметричная, сложнее, чем троичная и к тому же не имеет перед ней никаких преимуществ.

Остаются только двоичная и троичная системы. Двоичная система всем известна и на текущий момент получила повсеместное распространение в мире (в том числе из-за простых способов физической реализации битов — сначала с использованием электронных ламп, потом с использованием транзисторов).

Но у троичной симметричной системы счисления есть пять основных преимуществ (по сравнению с двоичной):

1. Естественное представление отрицательных чисел за счёт того, что в троичной симметричной системе счисления присутствует отрицательная цифра (-).
2. Округление производится простым отбрасыванием младших разрядов (потому что цифры меньше, чем половина основания системы счисления). Это не только позволяет не использовать специальную операцию округления, но и повышает точность некоторых вычислений (например, умножение чисел в экспоненциальной записи).

3. Операция сравнения троичных чисел производится поразрядным сравнением чисел, начиная со старшего разряда.
4. Для логических операций доступно третье значение («возможно» или «не известно»), что более естественно для логических отношений.
5. Троичная система — самая экономичная среди целочисленных позиционных систем счисления. Под экономичностью подразумевается количество оборудования, необходимого для реализации машинного слова в системе с основанием p, в предположении, что ячейка памяти усложняется пропорционально p. Это определение можно понимать как отношение количества представимых чисел к сложности технической реализации.

Также если посмотреть на таблицу умножения в троичной симметричной системе, то окажется, что она не сильно сложнее двоичной (0 на любое число – 0, 1 – тоже число, -1 – инвертированное число)

А таблица сложения, получается в каком-то смысле проще, потому что в троичном полном (трехвходовом) троичном сумматоре перенос в следующий разряд происходит в 8 / 27 случаях, а в двоичном – в 4 / 8 случаях (то есть операций переноса в следующий разряд для троичных регистров требуется меньше).

Дональд Кнут, известный американский учёный в области информатики и автор серии книг по алгоритмам и методам вычислительной математики, охарактеризовал троичную симметричную систему как «быть может самую изящную» и дал следующее предсказание относительно использования троичной симметричной системы счисления: «Возможно, симметричные свойства и простая арифметика этой системы окажутся в один прекрасный день весьма существенными» [8].

1.3 Исследования физической реализации троичных ячеек

Самое главное преимущество двоичной системы – это простота физической реализации (есть сигнал – нет сигнала). Это же одновременно является главным недостатком троичной системы.

В «Сетуни» (как и в нашем эмуляторе) троичная ячейка представлялась, как пара двоичных ячеек. Это приводит к неэффективному использованию оборудования (четвертое состояние битов не используется).

На протяжении последних десятилетий активно ведутся исследования в области разработки технологий для физической реализации ячеек с множественными устойчивыми состояниями.

В мае 2023 года была признана и зарегистрирована новая ячейка памяти с тремя устойчивыми состояниями [9]. Учёные Саратовского университета создали Т-образную углеродную нанотрубку. Внутри которой находится свободный фуллереен С60 [10] – молекула из 60 атомов углерода, расположенных в форме футбольного мяча. Эта молекула обладает уникальными свойствами: высокой плотностью, жёсткостью и устойчивостью к воздействию химических и физических факторов. Фуллерен способен перемещаться внутри нанотрубки под воздействием внешнего электрического поля и занимать три устойчивых положения в концах трубки.

2. Эмулятор

Нами разработан эмулятор троичного компьютера (виртуальная машина) с использованием языка программирования C++, на базе двоичного 64-битного компьютера (архитектуры x86-64).

Общий объём кода эмулятора составляет примерно 1500 строк.

В программе используются такие следующие классы:

- CPU – процессор,
- RAM – оперативная память,
- Trit – трит,
- Word – машинное слово из 32 тритов.

Полный код эмулятора доступен здесь [11].

2.1 Характеристики виртуальной машины

- Устройство памяти:
  - Триты хранятся как два бита (00 = 0, 01 = -1, 10 = 1)
  - Слова состоят из 32 тритов (64 бита)
  - Оперативная память содержит 3^8 = 6561 слов (адрес содержит 8 тритов)
- Процессор:
  - Содержит 7 регистров:
    - S, A – 2 для операции сложения (32 трита)
    - M, F – 2 для умножения/деления (32 трита)
    - K – выполняемая команда (адрес и код операции) (12 тритов)
    - C – счетчик команд (адрес выполняемой команды) (8 тритов)
    - P – указатель на вершину стека (8 тритов)
    - Также есть один флаг, вырабатываемый для каждой команды + (обычно в зависимости от знака числа) (1 трит)

- Количество операций — 38
  - 4 для переноса слова из оперативной памяти в регистр
  - 4 для переноса из регистра в оперативную память
  - 9 для работы с локальными переменными
  - 4 перехода (3 условных и 1 безусловный)
  - 4 операции (сложение, вычитание, умножение)
  - 2 операции инвертирования регистров (S или A)
  - 3 для работы с указателем на вершину стека
  - 3 обеспечивают работу функций
  - 1 для обмена значений регистров S, A
  - 4 для ввода, вывода

В нашем эмуляторе триты представляются, как два последовательно расположенных бита. Но известен пример программной реализации троичной виртуальной машины [12], в которой для эмуляции троичных слов используются пара двоичных слов такой же длины {P, M}, причём в первом слове P единицы стоят на тех местах, где в представлении троичного числа стоят +1, а в слове M — там, где в троичном слове стоят -1. Например, троичное число «00+0-0+-+» в используемом для эмуляции двоичном компьютере будет выглядеть, как {P, M} = {001000101, 000010010}. Такое представление по мнению разработчиков допускает параллельную обработку целых машинных слов и менее громоздко в программировании.

2.2 Устройство виртуальной машины

Как уже было сказано ранее, в процессоре эмулятора используются 38 операций, это довольно много, по сравнению, например, с ЭВМ «Сетунь», где было всего 24 команды. Это объясняется тем, что в реализованной нами машине отсутствует контроллер оперативной памяти.

В эмуляторе присутствует поддержка работы со стеком на уровне процессора, то есть в процессоре присутствует регистр для указателя на

вершину стека, а также машинные команды для инкремента, декремента и записи адреса в регистр P.

Команды начинают выполняться с адреса «--------», то есть регистр C перед началом выполнения команд равен «--------». Стек же растёт в обратную сторону, начиная с адреса «++++++++».

Архитектура с малым количеством инструкций процессора и аппаратной поддержкой стека, называют MISC – Minimum Instruction Set Computer [13].

2.3 Ввод информации

Можно вводить данные и инструкции вручную - присваивать нужной ячейке оперативной памяти значение. Но для этого нужно знать внутреннее строение эмулятора (его архитектуру), а также код каждой операции.

Это довольно неудобно, поэтому часто используют легко запоминающиеся буквенные команды, которые почти всегда транслируются в одну машинную команду. То есть это машинные команды, написанные на более понятном для человека языке. Этот язык программирования называется ассемблером – это представление команд процессора в виде, доступном для чтения человеком.

В ассемблере для нашего эмулятора есть 18 команд, не считая функции help:

- 2 для обмена данных между процессором и оперативной памятью
  1) snd
  2) mov
- 3 операции (сложение, вычитание, умножение)
  1) sum
  2) sub
  3) mul
- создание глобальной переменной – glb

- создание и удаление локальной переменной
  1) var
  2) del
- ввод и вывод из файла
  1) inp
  2) out
- обмен значений регистров S и A – swp
- инвертирование регистра – inv
- создание метки или функции – lbl
- вызов и возвращение из функции
  1) cll
  2) ret
- оператор перехода (передачи управления) – jmp
- команда присваивания ячейке памяти указанного значения – set
- команда завершения работы программы – end

2.4Тестирование

Пример программы на языке ассемблера – рекурсивное вычисления числа Фибоначчи, номер которого указан в файле “input.txt”:

`glb f, 0`	определение глобальной переменной f
`inp f`	запись значения из файла ("input.txt") в переменную f
`cll F(f)`	вызов функции F, с параметром f
`end`	окончание выполнения программы
	(пустые строчки в программе ставятся только после всей программы)
`lbl F(x)`	функция F
`snd S, x`	посылка параметра x в регистр S
`jmp 0, null`	если флаг состояния равен 0 (если отправленное в регистр S значения равно 0), то перейти на метку null (условный переход)

`var a, 0`	определение локальной переменной a
`mov A, a`	посылка значение регистра A в локальную переменную a
`glb 1, +`	определение глобальной переменной 1
`snd A, 1`	посылка значения переменной 1 в регистр A
`sub S, A`	вычитание регистра S, регистр A (результат в регистре S)
`jmp 0, one`	если полученное значения в регистре S равно нулю то перейти на метку one (условный переход)
`jmp def`	перейти на метку def (безусловный переход)
`lbl null()`	метка null
`del`	удаление параметра x с вершины стека
`ret`	возврат из функции F (т.к. ответ (0), уже находится в регистре S)
`lbl one()`	метка one
`snd S, 1`	посылка значения переменной 1 в регистр S
`snd A, a`	посылка значение локальной переменной a в регистр A
`del`	удаление локальной переменной a с вершины стека
`del`	удаление параметра x с вершины стека
`ret`	возвращение из функции F
`lbl def()`	метка def (default)
`var q, 0`	определение локальной переменной q
`mov S, q`	посылка значение регистра S в локальную переменную q
`swp`	обмен значений регистров S и A
`cll F(q)`	вызов функции F, с параметром q
`mov S, q`	посылка значение регистра S в локальную переменную q
`swp`	обмен значений регистров S и A
`snd A, 1`	посылка значения переменной 1 в регистр A
`sub S, A`	вычитание регистра S, регистр A (результат в регистре S)
`snd A, q`	посылка значения переменной q в регистр A
`mov S, q`	посылка значение регистра S в локальную переменную q
`cll F(q)`	вызов функции F, с параметром q
`sum`	сложить регистры S и A (результат в регистре S)
`snd A, a`	посылка значение локальной переменной a в регистр A
`del`	удаление переменной q с вершины стека
`del`	удаление переменной a с вершины стека
`del`	удаление параметра x с вершины стека
`ret`	возврат из функции F

Этот код находится в файле 'program.txt' и по умолчанию запускается для числа 28 (записано в файле 'input.txt', примерное время работы 9 секунд). Ответ будет записан в регистре S. Для ускорения работы функций, возвращаемое значение можно записывать в регистр.

Для того, чтобы начать работу эмулятора, нужно в окне командной строки ввести цифру -1 (это будет означать, что нужно выполнить все инструкции, не останавливаясь). После завершения работы эмулятора, в командной строке выведется состояние каждого регистра. В регистре S будет записано число Фибоначчи.

```
S - 00000000000000000000+--+0++00--+0 317811
A - 00000000000000000000000000000000 0
C - ------00++++++++++++++++++++++++
K - 00000000000000000000000000000000
M - 00000000000000000000000000000000 0
F - 00000000000000000000000000000000 0
P - ++++++00++++++++++++++++++++++++
W - 0
```

Заключение

В этой статье мы рассказали о преимуществах и недостатках троичных компьютерах и о реализации настоящей троичной машины, на примере эмулятора.

В процессе написания эмулятора и этой статьи автор узнал:

- устройство и двоичных и троичных компьютеров,
- синтаксис языка ассемблера,
- возможные конструкции троичных ячеек,
- принцип работы ЭВМ «Сетунь» и ее архитектуру.

В процессе работы были выполнены все поставленные задачи:

1. Изучены преимущества разных систем счисления.
2. Изучена литература по устройству троичного и двоичного компьютера.
3. Спроектирована архитектура троичного компьютера.
4. Написан код эмулятора.
5. Придуман удобный способ ввода информации в эмулятор — ассемблер.
6. Написаны тестовые программы для эмулятора – рекурсивное вычисление чисел Фибоначчи.

Самым трудным в работе над созданием эмулятора – была реализация вызова функций в языке ассемблера.

Список литературы

[1] Закон Мура / [Электронный ресурс] // Википедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/Закон_Мура

[2] John L. Hennessy, David A. Patterson. Computer Architecture: A Quantitative Approach, Sixth Edition // Morgan Kaufmann Publishers Inc., San Francisco, CA, United States, 2017 // ISBN 978-0-12-811905-1 - стр. 3 – URL: https://acs.pub.ro/~cpop/SMPA/Computer%20Architecture,%20Sixth%20Edition_%20A%20Quantitative%20Approach%20(%20PDFDrive%20).pdf

[3] Владимирова Ю.С. Введение в троичную информатику. – Москва, АРГКМАКМЕДИА, 2015.

[4] Задача о гирях / [Электронный ресурс] // Википедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/Фибоначчи#Задачи_о_гирях

[5] The ternary calculating machine of Thomas Fowler / [Электронный ресурс] // : [сайт]. – URL: https://www.mortati.com/glusker/fowler/index.htm

[6] Сетунь(компьютер) / [Электронный ресурс] // Википедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/Сетунь_(компьютер)

[7] Брусенцов Н.П., Жоголев Е.А., Маслов С.П. Общая характеристика малой цифровой машины «Сетунь-70» // Вычислительная техника и вопросы кибернетики. Вып.10. - Л.: Изд-во Ленингр. ун-та, 1974. С.3-20.

[8] Кнут Д. Искусство программирования. Т.2. Получисленные алгоритмы. – Москва: Издательский дом «Вильямс», 2007.

[9] Новая троичная ячейка памяти / [Электронный ресурс] // indicator.ru : [сайт]. – URL: https://indicator.ru/mathematics/novaya-troichnaya-yacheika-pamyati-iz-fullerenai-nanotrubki-povysit-proizvoditelnost-sovremennykh-processorov-27-07-2023.htm

[10] Фуллерен / [Электронный ресурс] // Википедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/Фуллерен

[11] [Электронный ресурс] // [сайт]. – URL: https://github.com/VolynetsPR/Ternary

[12] Бурцев А.А, Сидоров С. А. Троичная виртуальная машина и троичная ДССП / [Электронный ресурс] // Сборник тезисов докладов Национального Суперкомпьютерного Форума (НСКФ), 2014: [сайт]. – URL: https://2014.nscf.ru/TesisAll/0_PostMoore_Plenar/06_208_BurtsevAA.pdf

[13] Центральный процессор / [Электронный ресурс] // Википедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/Центральный_процессор

Приложение

«Сетунь-70»

Пульт «Сетуни-70»

Схема пульта «Сетуни» 1965 года (идентичен пульту 1958)

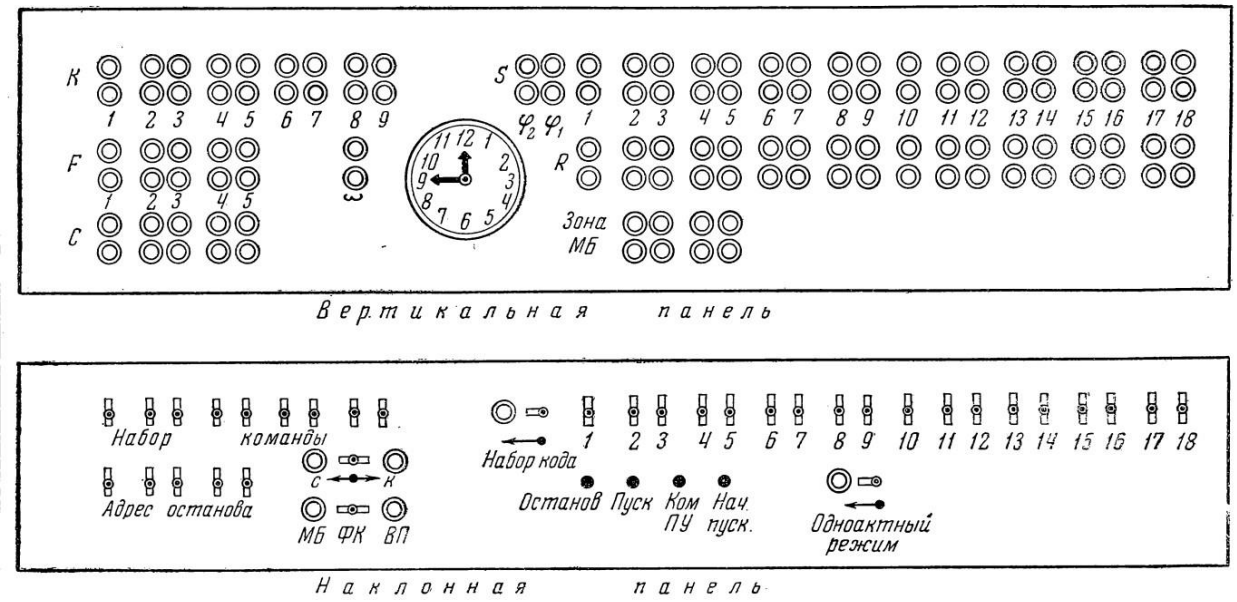

Рис. 3. Пульт управления

Пульт «Сетуни» 1958 года