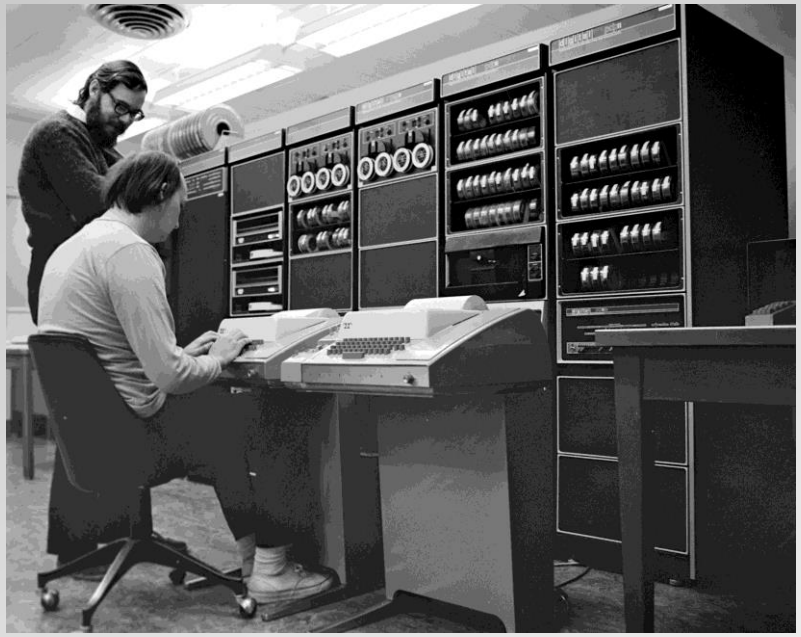
# Программирование на языках ассемблера и C для архитектуры PDP-11 на примере компьютера БК-0010

Часть 1

# Ответы на вопросы

**Что такое PDP-11?**

PDP-11 – это компьютерная архитектура, разработанная компанией DEC в 1970 году. В скором времени компьютеры на PDP-11 стали очень популярны в науке и промышленности, так как были хоть и слабее, но гораздо компактней и дешевле мейнфреймов от IBM. Именно на этих компьютерах впервые широко распространилась ОС Unix, для них был создан язык программирования C. И, хотя уже к началу восьмидесятых архитектура начала устаревать, DEC выпускала машины на PDP-11 вплоть до 1990 года, делая их все более производительными и уменьшая их габариты



Деннис Ритчи и Кен Томпсон перед одной из ранних моделей DEC PDP-11

Одна из поздних моделей DEC PDP-11

# Ответы на вопросы

**Но ведь писать программы под промышленные компьютеры без графики скучно**

В то время, как за рубежом архитектура PDP-11 использовалась лишь в «больших» компьютерах для нужд науки и бизнеса, в СССР серийно выпускались и домашние машины на этой архитектуре. Наибольшее распространение получили модели:

1. БК-0010;
2. БК-0011М;
3. УКНЦ.

Менее известны такие компьютеры как:

1. Союз-Неон;
2. МС-0515;
3. Немига;
4. МК-90.

Эти машины имеют разный форм-фактор, производительность и графические возможности, но их объединяет общая архитектура, поэтому, научившись писать программы под одну из них, можно довольно быстро освоить и другую.

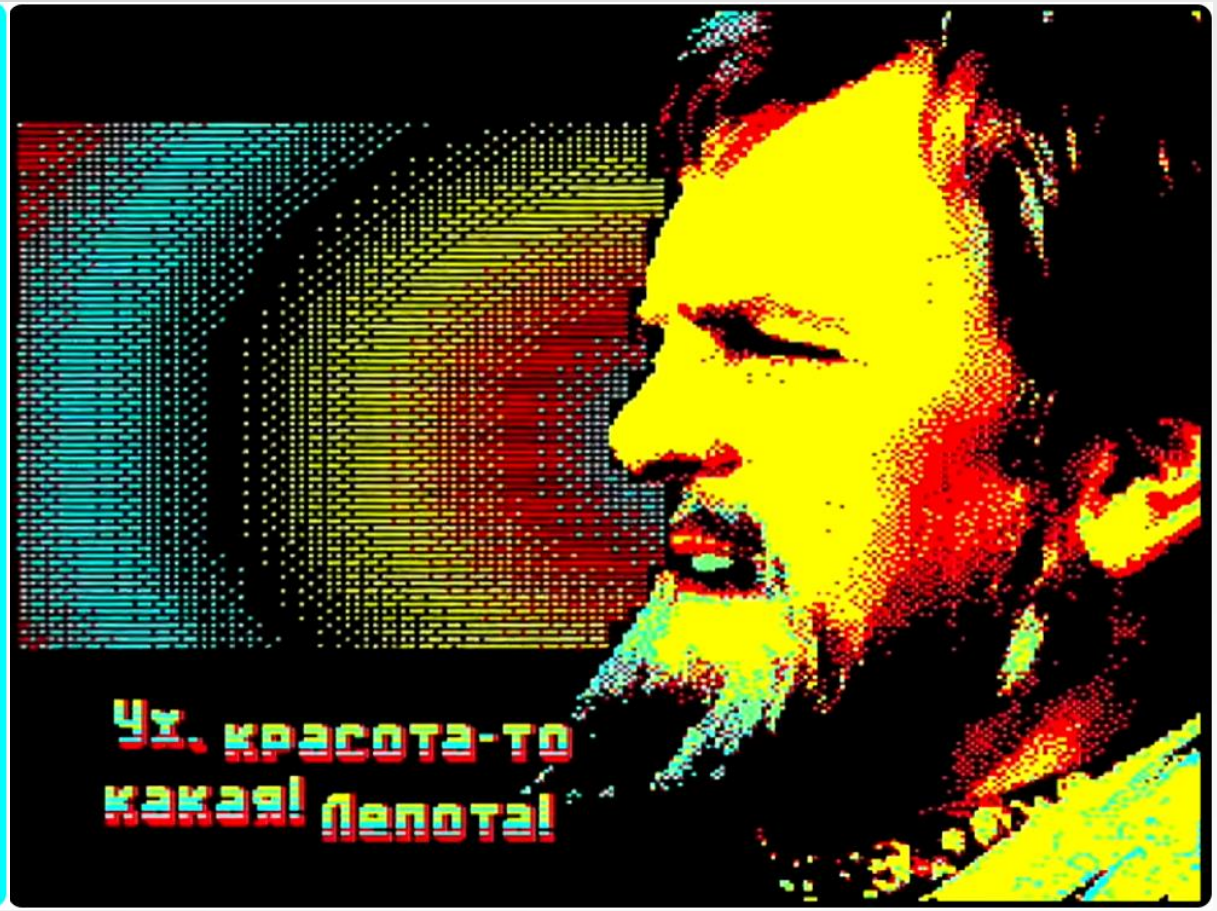

Некоторые персональные компьютеры на архитектуре PDP-11

# Ответы на вопросы

**На что способны эти машины?**

Как уже говорилось, у разных компьютеров разные графические возможности. Приведём несколько примеров, того, что можно «выжать» из этих «старичков»

«Bad apple» для БК-0011м от Manwe




«Однажды» для БК-0011м от Excess team

# Ответы на вопросы

**На что способны эти машины?**


Демонстрация Союз-Неон


«Весёлая птичка» для Электроника МК-90 от azya

# Ответы на вопросы

**Это сложно?**

Архитектура PDP-11 имеет один из самый дружелюбных языков ассемблера. Благодаря этому она и обрела большую популярность.

Данное руководство познакомит вас с языком ассемблера PDP-11 и научит компилировать код на языке C с помощью компилятора GCC так, чтобы он запускался на этой архитектуре. Наличие языка C упростит нам жизнь, но, к сожалению, GCC не в полной мере поддерживает PDP-11, поэтому без знания ассемблера нам не обойтись.

Освоив материал, изложенный в этих видео, мы сможете писать простые текстовые и графические программы для БК-0010. Затем, уже самостоятельно, вы сможете быстро освоить более совершенный компьютер БК-0011м. Впоследствии можно перейти к другим машинам, например Союз-Неон, УКНЦ или МК-90.


БК-0010

# Ответы на вопросы

**Мне придётся изучать все самостоятельно?**

Нет, существует большое сообщество программистов, пишущих код под различные компьютеры, совместимые с PDP-11. На момент написания руководства, в крупнейшем telegram канале, посвящённом БК-0010 и БК-0011м состоит чуть менее пяти сотен человек. Там можно попросить помощи, изучить чужой опыт, скачать литературу по теме. Ссылка на канал дана в QR коде

Ежегодно проводится несколько фестивалей, посвящённых ретро компьютерам, где устраиваются конкурсы программ (так называемых «демо»). Крупнейшие фестивали это

1. Демодуляция
2. Cafe Computer Festival

Telegram канал

Демодуляция 2023

Демодуляция 2023

# Ответы на вопросы

**Почему именно БК-0010?**

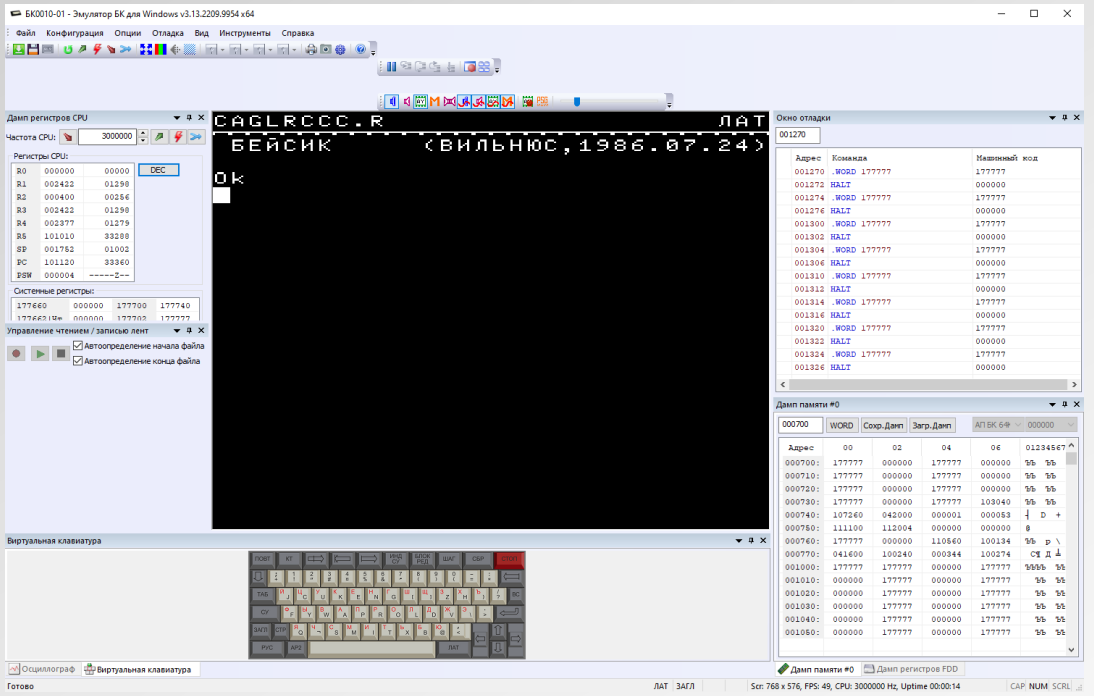
БК-0010 считается самым понятным из советских компьютеров с архитектурой PDP-11. Так, в нем довольно просто устроена память и графическая часть. Кроме того, это самый распространённый в наших краях компьютер на архитектуре PDP-11.

**Какие знания требуются, чтобы начать?**

Для освоения материала, нужны знания из школьного курса информатики, желательно знания языка программирования C.

**Нужно ли мне купить БК-0010?**

Нет, так как существует несколько эмуляторов этого компьютера. В данном руководстве мы будем пользоваться Gid, который можно скачать с сайта разработчика.




Эмулятор Gid

# Хранение чисел со знаком. Дополнительный код

Процессор с архитектурой PDP-11 способен оперировать с двумя типами данных:

1. числа без знака;
2. числа со знаком.

Поддержка чисел с плавающей точкой пришла с появлением расширенного набора инструкций, и доступна далеко не во всех компьютерах. В частности, в БК-0010 её нет. Таким образом, для разработки программ под PDP-11 нам нужно понимать, как хранятся в памяти компьютера числа без знака и числа со знаком.

# Хранение чисел со знаком. Дополнительный код

Вы наверняка знаете как устроена двоичная система счисления, и как переводятся числа без знака из десятичной системы в двоичную и обратно. Однако, числа со знаком представляются в двоичной системе счисления совершенно иначе. Сначала приведём пример сложения четырёхразрядных чисел:

$$\begin{array}{rr} 0001 & 1 \\ +0011 & +3 \\ 0100 & 4 \end{array}$$

мы получили ожидаемый результат.

Теперь вычитание:

$$\begin{array}{rr} 0001 & 1 \\ -0011 & -3 \\ 1110 & -2 \end{array}$$

нам потребовалось занимать из старших разрядов уменьшаемого, как при обычном вычитании в столбик. При этом мы считали, что уменьшаемое имеет неограниченную разрядность, и все разряды слева от четвёртого равны единице. Таким образом, мы получили число 1110, но мы трактуем его не как 14, а как -2. Почему так?

# Хранение чисел со знаком. Дополнительный код

Дело в том, что старший разряд двоичного числа со знаком называется знаковым. Если число положительное, то он равен нулю, а если отрицательное – единице. Значит число 1110 – отрицательное. Но почему оставшиеся разряды 110 считаются двойкой? Это и есть дополнительный код.

Попробуем выполнить ещё два действия с числом -2, чтобы проверить его свойства:

$$\begin{array}{rl} 0101 & 5 \\ +1110 & +(-2) \\ 0011 & 3 \end{array}$$

этот пример доказывает, что при сложении с положительными числами действительно 1110 ведёт себя как -2.

# Хранение чисел со знаком. Дополнительный код

Рассмотрим второй пример:

$$\begin{array}{rr} 1110 & -2 \\ +1110 & +(-2) \\ 1100 & -4 \end{array}$$

Но действительно ли 1100 – это число -4? Проверим это:

$$\begin{array}{rr} 0000 & 0 \\ -0100 & -4 \\ 1100 & -4 \end{array}$$

можно видеть, что 1100 – это и в правду -4. В этом и заключается смысл дополнительного кода: мы можем не задумываться о знаках при операциях с положительными и отрицательными числами, они получаются автоматически.

# Хранение чисел со знаком. Дополнительный код

Как получается дополнительный код? Чтобы получить отрицательное число в дополнительном коде, нужно вычесть равное по абсолютной величине положительное число из нуля. Поэтому код и называется дополнительным, ведь отрицательное и положительное числа, равные по модулю, дополняют друг друга до нуля или, как говорят, до переноса в разряд за пределы числа.

Есть более простое правило получения отрицательного числа из положительного в уме: нужно инвертировать исходное число, то есть заменить единицы нулями, а нули – единицами, и прибавить к получившемуся числу единицу. Приведём пример для числа 4:

$0100 \rightarrow 1011 \rightarrow 1100$

получили -4.

Можно произвести и обратное преобразование, выполнив шаги в противоположном порядке: вычесть единицу, а затем инвертировать полученное число.

# Хранение чисел со знаком. Дополнительный код

Приведём пример, когда дополнительный код не работает:

$$\begin{array}{rr} 0110 & 6 \\ +0011 & +3 \\ 1001 & -7 \end{array}$$

Дело в том, что произошло так называемое переполнение, то есть перенос единицы в знаковый разряд. Это произошло, потому что заданная разрядность оказалась мала для сложения двух предложенных чисел. Если мы добавим ещё один разряд слева и будем считать его знаковым, система будет работать

$$\begin{array}{rr} 00110 & 6 \\ +00011 & +3 \\ 01001 & 9 \end{array}$$

# Хранение чисел со знаком. Дополнительный код

Процессор с архитектурой PDP-11 способен оперировать 8- и 16-разрядными числами. Поэтому, знаковый разряд здесь фиксирован – это бит 7 для байта и 15 для слова (двух соседних байтов). Что же произойдёт при переполнении байта или слова? Процессор отмечает переполнение и записывает единицу в специальный регистр. Мы можем проверить значение этого регистра, узнать, что возникла ошибка и отреагировать на неё тем или иным образом.

Приведём диапазоны допустимых значений для 8- и 16-разрядных чисел:

| | **8 бит** | **16 бит** |
|---|---|---|
| **беззнаковое** | 0…255 | 0…65535 |
| **знаковое** | -128…127 | -32768…32767 |

# Хранение чисел со знаком. Дополнительный код

Важно отметить, что существует два варианта операции сдвига. Первый тип называют логическим сдвигом влево(вправо). В этом случае двоичная запись числа переносится на один разряд левее(правее), причём разряд, вышедший за пределы байта или слова, сохранятся в специальный регистр процессора, а в освободившийся бит записывается нуль. Приведём пример:

01110111 – исходное число, равное 119

00111011 – исходное число после логического сдвига право, получили 59

11101110 – исходное число после логического сдвига влево, получили 238

Можно видеть, что логический сдвиг позволяет делить умножать числа без знака на два. Результат деления при этом округляется в меньшую сторону, но, как говорилось выше, при сдвиге вправо содержимое младшего разряда будет записано в специальный регистр процессора, и при желании можно будет выполнить коррекцию погрешности.

# Хранение чисел со знаком. Дополнительный код

Однако, если мы выполним логический сдвиг над числом со знаком, мы можем получить неправильный результат, так как может измениться знаковый разряд. Например:

10000111 – исходное число, равное -57

01100011 – исходное число после логического сдвига вправо, получили 99

10001110 – исходное число после логического сдвига влево, получили -114.

# Хранение чисел со знаком. Дополнительный код

Для чисел со знаком предназначен второй вариант сдвига – арифметический. В этом случае сдвиг влево работает, как и при логическом сдвиге, а сдвиг вправо сохраняет значение в знаковом разряде. Вернёмся к прошлому примеру:
10000111 – исходное число, равное -57
11100011 – исходное число после арифметического сдвига вправо, получили -29
10001110 – исходное число после арифметического сдвига влево, получили -114.

Как и раньше, результат деления округляется, но это так же можно парировать, если следить за соответствующим регистром процессора.

# Адресное пространство PDP-11

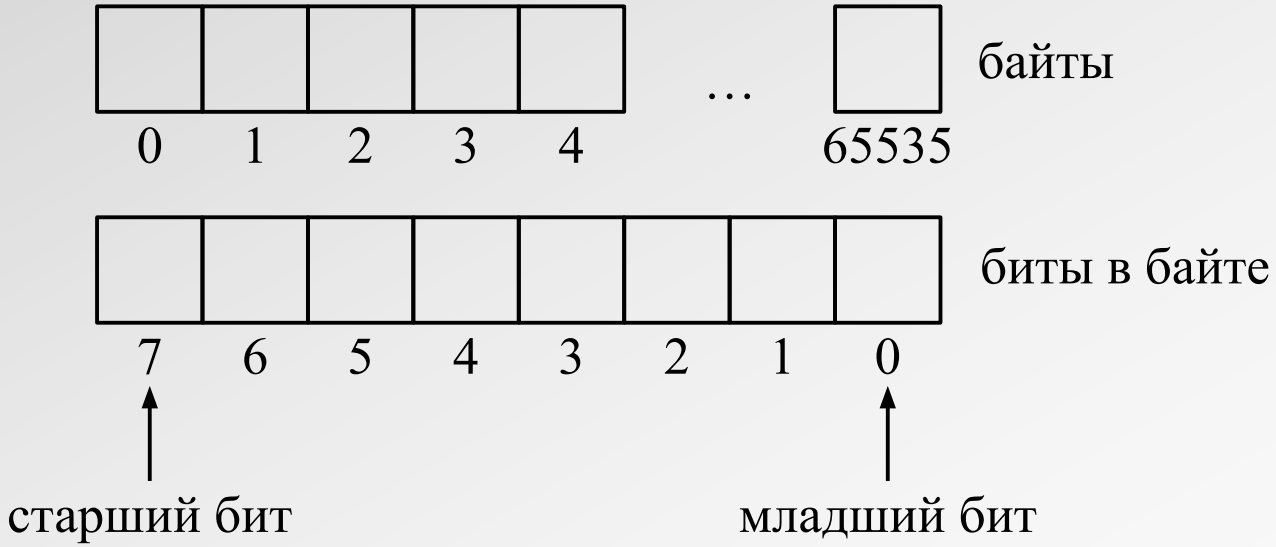
Архитектура PDP-11 является шестнадцатиразрядной. Это означает, что центральный процессор компьютера оперирует числами, состоящими из шестнадцати бит, то есть двух байт. Так как PDP-11 использует единую шину, то это ограничение распространяется и на адреса, и на данные. Следовательно, без дополнительных доработок, компьютер с архитектурой PDP-11 способен адресовать $2^{16} = 65536$ байт.

# Адресное пространство PDP-11

В компьютерах на архитектуре PDP-11 существует два способа работы с памятью:

1. по байтам;
2. по словам.

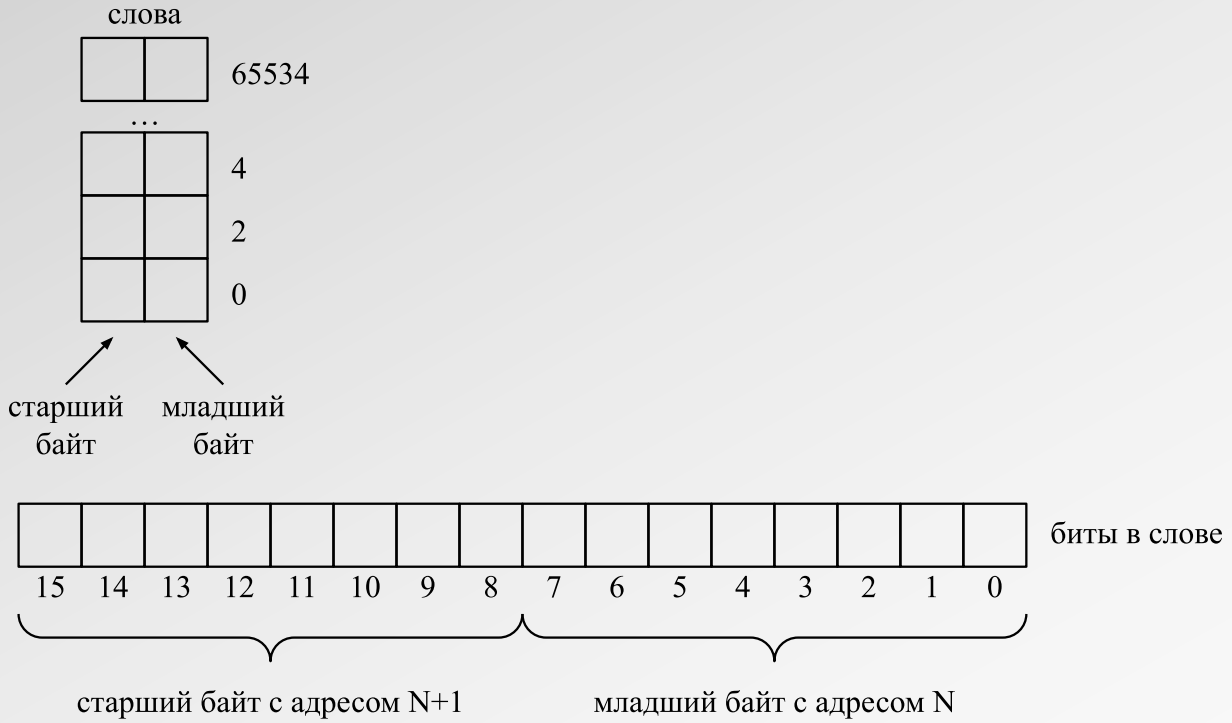
В первом случае память представляется в виде массива байтов. Каждый байт имеет индекс, который называют адресом. Биты внутри байта также перенумерованы, но эта нумерация идёт справа налево. Нулевой бит называют младшим, а седьмой – старшим.



# Адресное пространство PDP-11

Словом называется пара байтов с адресами $N + 1$ и $N$, где $N$ – чётное число. Байт с адресом $N$ называют младшим, а с $N + 1$ – старшим. Адрес слова совпадает с адресом младшего байта и поэтому всегда является чётным числом.

Таким образом, длина слова равняется шестнадцати битам. Нумерация битов в слове, как и в байте, идёт справа налево. Нулевой бит называют младшим, а пятнадцатый – старшим.



# Адресное пространство PDP-11

Можно видеть, что для  битов, принадлежащих чётному байту, нумерация битов в рамках байта совпадает с нумерацией рамках слова. А для бит, принадлежащих нечётным байтам, нумерации отличаются.

# Адресное пространство PDP-11

Когда мы работаем с данными по нечётному адресу, мы манипулируем отдельным байтом. Однако, когда мы обращаемся к чётному адресу, мы можем получить доступ как к байту, так и к целому слову. Как избежать путаницы? Дальше мы увидим, что среди команд PDP-11 есть те, которые работают с отдельным байтом, и те, которые работают с целым словом.

# Адресное пространство PDP-11

Ещё одной важной традицией среди программистов PDP-11 является запись адресов в восьмеричной системе исчисления, в которой используются только цифры от 0 до 7. Здесь и далее ноль, стоящий в начале числа, будет указывать на то, что оно записано в восьмеричной системе исчисления. Таким образом, адреса принадлежат диапазону от 0 до 0177777.

# Распределение адресного пространства на БК-0010

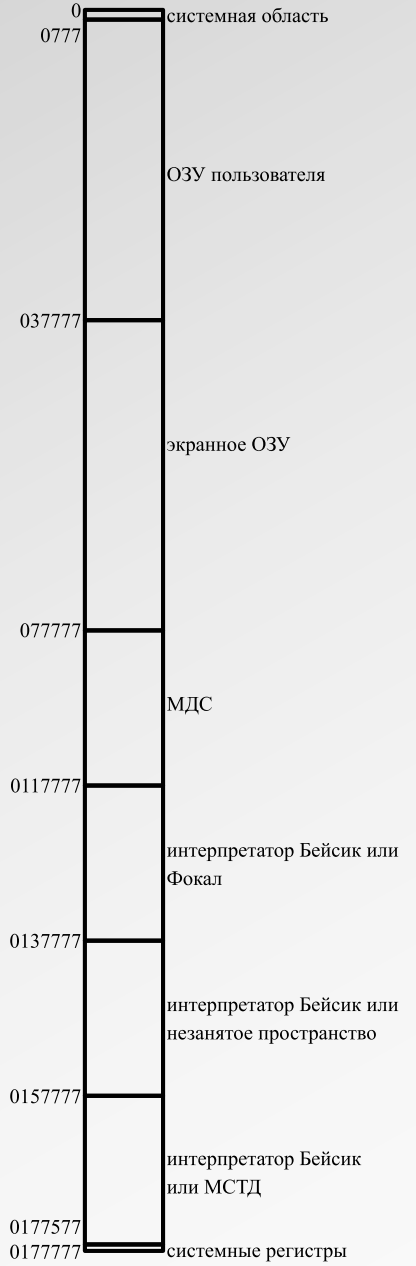
Покажем, какие данные расположены в адресном пространстве БК-0010. Заметим, что на других компьютерах их расположение может отличаться.

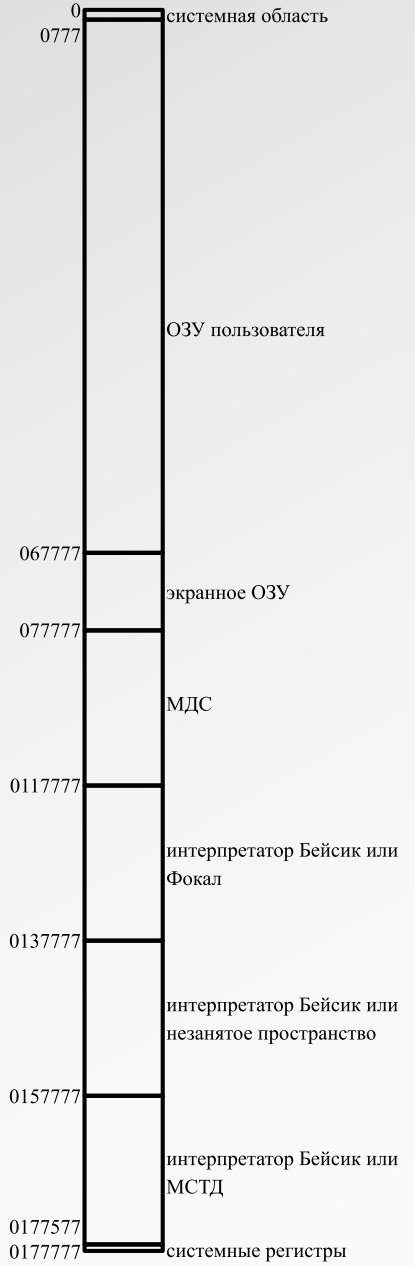
| Диапазон адресов | Объем | Возможно чтение | Возможна запись | Описание |
|---|---|---|---|---|
| 0…0777 | 0.5Кб | да | да | системная область и стек |
| 01000...037777 или 01000...067777 | 15.5Кб или 27.5Кб | да | да | ОЗУ пользователя |
| 040000...077777 или 070000...077777 | 16Кб или 4Кб | да | да | экранное ОЗУ |
| 0100000...0117777 | 8Кб | да | нет | МДС |
| 0120000...0137777 | 8Кб | да | нет | интерпретатор Бейсик или Фокал |
| 0140000...0157777 | 8Кб | да | нет | интерпретатор Бейсик или незанятое пространство |
| 0160000...0177577 | 7.87Кб | да | нет | интерпретатор Бейсик или МСТД |
| 0177600...0177777 | 0.13Кб | да | да | системные регистры |

# Распределение адресного пространства на БК-0010

Кратко опишем каждую из областей:

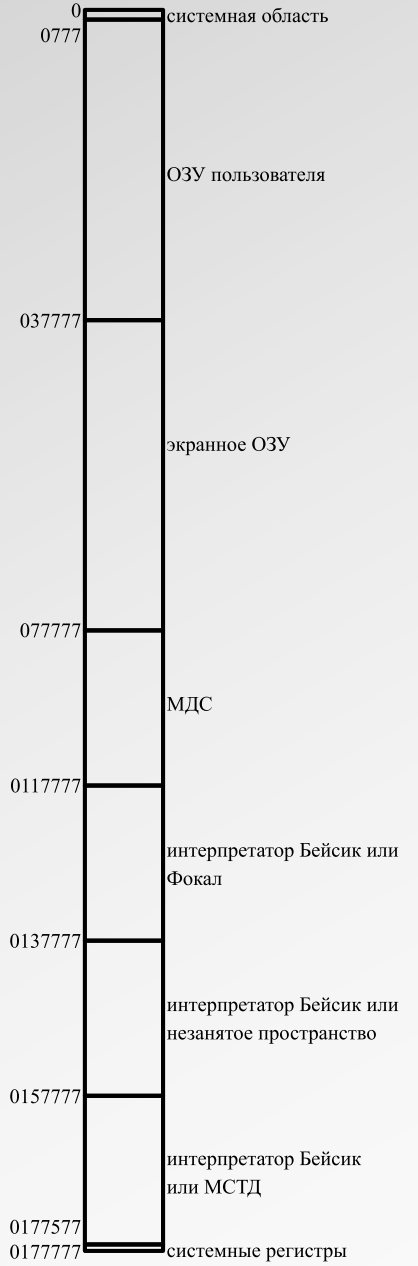
- Системная область и стек используются для нужд самого компьютера. Об этом будет подробней рассказано далее;
- ОЗУ пользователя — область, в которой программист может разместить свою программу и данные. Этот раздел можно увеличить на 12Кб за счёт экранного ОЗУ, но при этом уменьшается разрешение выводимого изображения. При включении режима расширенной памяти (РП) данные, лежащие в бывшем экранном ОЗУ, не уничтожаются. При обратном переключении, данные, лежавшие в области памяти, возвращённой экранному ОЗУ, уничтожаются;

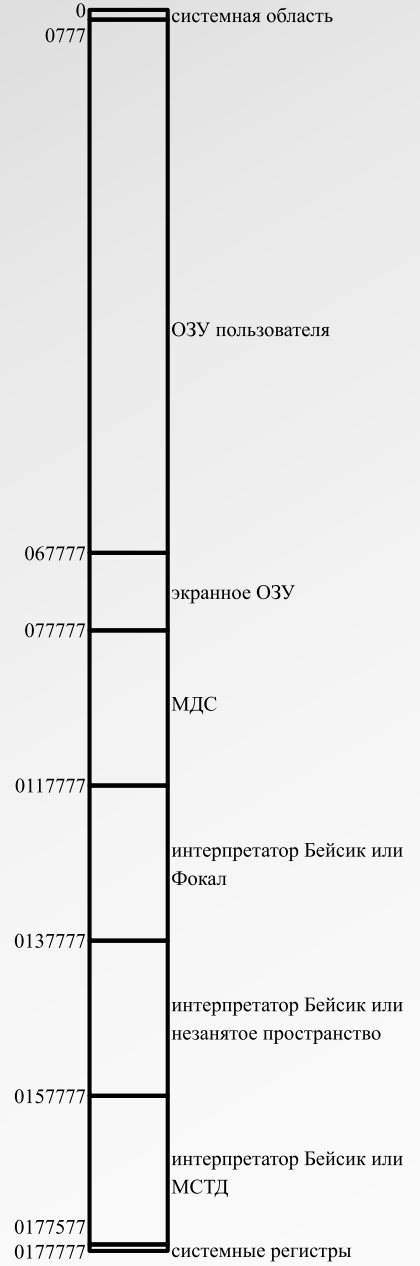


# Распределение адресного пространства на БК-0010

- Все данные, записанные в экранное ОЗУ, будут тут же отображены на мониторе и наоборот, любое изменение изображения, например печать нового символа, изменит содержимое этой области;
- Следующие области представляют собой микросхемы ПЗУ, установленные в компьютер, и содержат базовые программы. Запись данных по этим адресам невозможна. Существует два стандартных набора микросхем для БК-0010. Первый включает интерпретатор Бейсик, а второй – интерпретатор Фокал и мониторную систему тестов и диагностики (МСТД). Монитор-драйверная система (МДС) присутствует всегда;
- Системным регистрам будет посвящён отдельный раздел.



# Общее представление о том, как исполняет программу процессор с архитектурой PDP-11

Программа хранится в памяти компьютера в виде последовательность двоичных чисел. При её запуске процессор сохраняет адрес начала программы в специальном регистре, который называют счётчиком команд, и считывает первое слово программы, расположенное по этому адресу, после чего счётчик команд увеличивается на 2. В зависимости от того, какая команда прочитана, процессор может сразу выполнить её (если в ней заключены все необходимые данные), или заняться поиском указанных в команде данных.

Эти дополнительные данные могут находиться где-то в памяти, либо входить в состав самой команды, но не в первое её слово. Команда процессора с архитектурой PDP-11 может иметь длину до трёх слов, причём первое слово – это всегда код самой команды, а остальные – данные (например числа или адреса, по которым расположены числа). Если команда состоит более чем из одного слова, то содержимое счётчика команд в процессе её выполнения увеличивается на 4 или на 6, после чего процессор переходит к выполнению следующего шага программы. Таким образом, счётчик команд всегда содержит адрес следующей инструкции.

# Общее представление о том, как исполняет программу процессор с архитектурой PDP-11

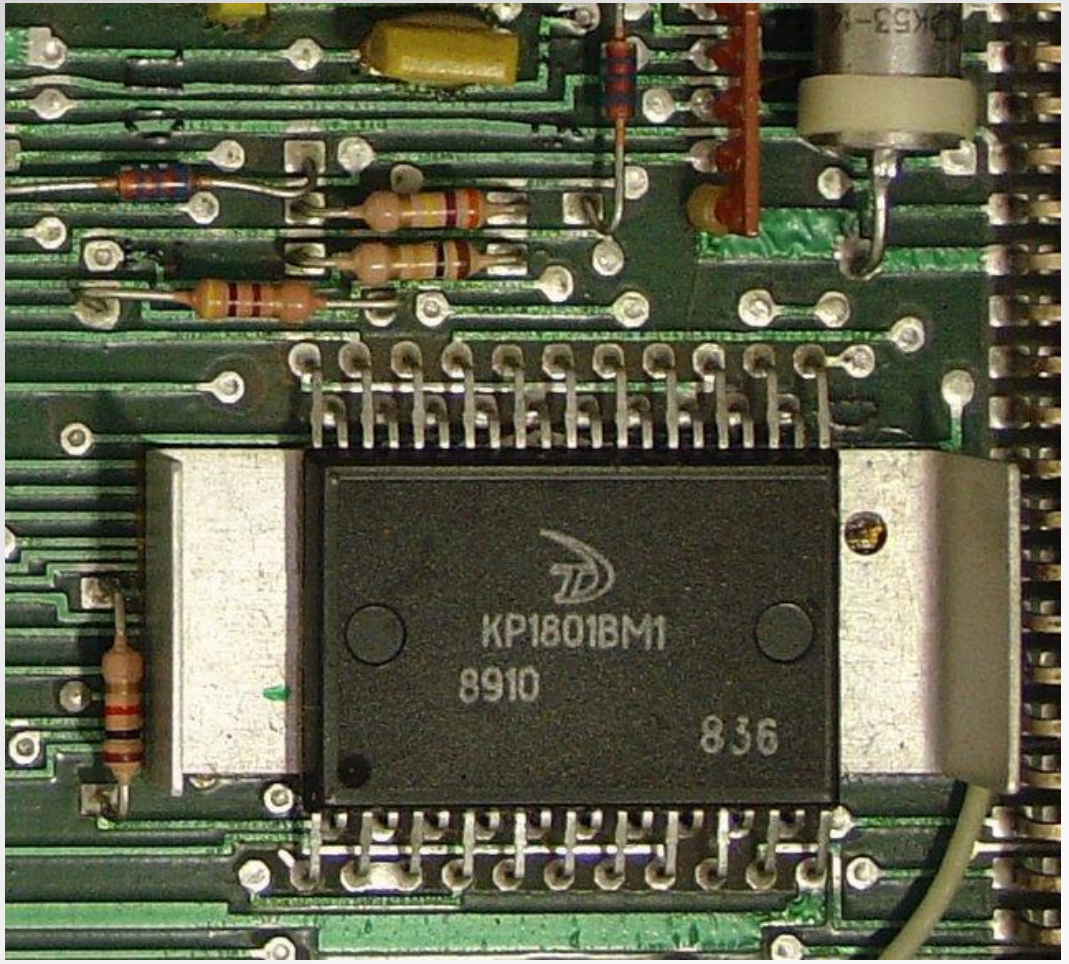
Можно видеть, что описанным выше способом, можно выполнять только линейную последовательность команд, когда шаги следуют строго один за другим. Однако, теоретиками доказано, что этого недостаточно для решения любого типа задач. Необходима также поддержка условий (то есть ветвления) и циклов. В архитектуре PDP-11 это решается очень просто – достаточно изменить значения счётчика команд, и процессор начнёт выполнять команды, расположенные уже в другом месте адресного пространства.

# Общее представление о том, как исполняет программу процессор с архитектурой PDP-11

Как устроено первое слово команды в архитектуре PDP-11? Не вдаваясь в подробности, можно сказать, что это слово разбито на части, состоящие из нескольких разрядов. Часть каждая такая часть описывает определенное свойство команды. Приведём эти свойства:

- Код команды;
- Типы операндов (отсутствует, константа, регистр процессора, адрес в памяти);
- Нужно ли выполнить действие над отдельным байтом или же над целым словом.

Более подробно о командах PDP-11 будет сказано далее.




Процессор БК-0010