1 Представление чисел с фиксированной точкой. Прямой, обратный и дополнительный код. Формирование битовых признаков переноса, переполнения, отрицательного результата, нуля

Представление чисел с фиксированной точкой. В работе. Мне нужно перезагрузиться в другую систему, дабы найти там материалы.

Прямой, обратный и дополнительный код. Есть три представления чисел с фиксированной точкой в ЭВМ:

- Прямой — модуль числа представляется, что для положительных, что и для отрицательных одинаково. Под знак числа отводится старший разряд. Область значений таких чисел симметрична, но ограничена тем, что у нас есть два представления нуля (положительный и отрицательный). Для N-разрядного целого двоичного числа область значений — $[-2^{N-1}; 2^{N-1}]$;

- Обратный код — представление положительных чисел сходно с прямым. Для представления отрицательных чисел используется инверсия положительного числа того же модуля. Для N-разрядных целых двоичных чисел область значений — $[-2^{N-1}+1; 2^{N-1}-1]$. Недостаток с двойным кодированием нуля также присутствует, но область значений также симметрична.

- Дополнительный код — представление положительных чисел также как и в прямом коде. Код отрицательных чисел формируется посредством инверсии всех разрядов положительного числа и прибавлении единицы. Область значений уже не симметрична (отрицательных чисел на одно больше, если ноль считать ни тем не другим), но исправляется проблема с двойным кодированием нуля. Для N-разрядной сетки двоичных целых чисел область значений $-[-2^{N-1}; 2^{N-1}-1]$. Обоснование формулы для N-разрядных:

$$\text{Отрицательное число это:} M' = 2^N - M, \text{где} \qquad M \geq 0$$
$$\text{Прибавим и вычтем из формулы 1:} M' = 2^N - 1 + 1 - M =$$
$$= ((2^N - 1) - M) + 1 = \bar{M} + 1$$

Формирование битовых признаков результата. Регистр флагов — регистр процессора (FLAGS), отражающий текущее состояние процессора. Регистр флагов содержит группу флагов состояния (арифметические флаги) и флаги управления. В БЭВМ регистром состояния является регистр PS (Program State) в его младших 4 разрядах хранятся флаги NZVC.

Перенос (C) CF (Carry Flag) — арифметический флаг переноса, в нем фиксируется перенос из старшего разряда при сложении и заем в старший разряд при вычитании. При умножении CF показывает возможность (=0) и невозможность (=1) представления произведения в том же формате, что и операндов. Флаг переноса является индикатором ошибки переполнения в беззнаковой арифметике. Используется в командах ветвления (условных переходах) для беззнаковой арифметики. В БЭВМ устанавливается по результату только тогда, когда открыт вентиль SETC, для этого используются 3 сигнала (выходящие из АЛУ): $_O$ (Carry old — флаг переноса до исполнения команды), C_N (Carry new — новый флаг, сформировавшийся после исполнения команды), C_{14} (Carry 14 — перенос в 14 разряд). На вход коммутатора пропускаются только 2 бита, связанные с переносом: C_N (Либо как C_N с АЛУ, либо как C_O в зависимости от установки вентилей) и C_{14}, использующийся для выставления флага переполнения.

Переполнение (V) OF (Overflow Flag) — флаг переполнения. Устанавливается в командах сложения и вычитания, если результат не помещается в формате, при этом и операнды и результат интерпретируются как знаковые числа. Аппаратно он формируется совпадением переносов из двух старших разрядов при сложении и заемов в два старших разряда при вычитании (если они совпадают, то флаг равен 0). Переполнение фиксируется 3 способами:

- сравнение знаков операндов и суммы: если знак суммы отличается от знаков операндов, то фиксируется переполнение;

- сравнение переносов из двух старших разрядов: если они не совпадают, то фиксируется переполнение;
- использование модифицированного знака (под знак отводится два разряда, второй дублирует знак).

В БЭВМ флаг переполнения помимо того, что он выставляется лишь при открытом вентиле SETV, использует биты C_N и C_{14} по следующей формуле: $V = C_N \oplus C_{14}$.

Отрицательный результат (N) SF (Sign Flag) — флаг знака, в котором копируется старший разряд результата. В БЭВМ копируется именно 15 бит результата при открытом вентиле STNZ.

Флаг нуля (Z) ZF (Zero Flag) — флаг нуля, устанавливается при нулевом значении результата, в противном случае сбрасывается. В БЭВМ устанавливается с помощью 16 входового элемента ИЛИ-НЕ (NOR) при открытом вентиле STNZ.

2 Базовые элементы вычислительной техники: ячейки, регистры, шины, вентили, тактовые генераторы, логические схемы, триггеры, счетчики, сумматоры

Ячейки Для хранения информации в ЭВМ используются ячейки памяти двух видов:

1. SRAM — Static Random Access Memory (Используется в основном в ПЗУ и его видах). Работает на 6 транзисторах за счет положительно-обратной связи. Не требует постоянной подзарядки, данные хранятся и без нее.
2. DRAM — Dynamic Random Access Memory (Используется в основном в ОЗУ). Требует лишь один транзистор и конденсатор (можно без если транзистор сам имеет паразитную емкость). Требует постоянной подзарядки из-за разрядки конденсатора, иначе данные теряются.

Регистры

Шины Набор проводов для передачи информации между компонентами логических схем. Имеет разрядность передаваемой информации, указанную обычно косой чертой на шине с числом разрядов рядом. На своих концах требует установку заглушек, для предотвращения отражения сигнала.

Вентили Один из компонентов логических схем, предназначенный для пропускания или задержки сигнала. Имеет два входа (входной сигнал и управляющий) и один выход (выходной сигнал). По сути работает как элемент И при положительном кодировании. При отсутствии управляющего сигнала (0), не пропускает сигнал на выход (0), при наличии управляющего сигнала (1) — пропускает вход (1/0).

Тактовые генераторы

Защелки

Триггеры

Счетчики

Сумматоры

3 Операционная система Unix — ядро ОС и файловая система.

Базовые понятия операционных систем Операционная система — является исторической заменой операторов первых ЭВМ, в чьи обязанноти входило запись программ в память ЭВМ, запуск этих программ и выгрузка из памяти результатов работы различных программ. Программы, написанные программистами, загружались в память с помощью перфокарт и считывались также на перфокарты. Современные ОС отличаются от операторов тем, что могут регулировать работу ЭВМ, выполняя несколько процессов практически одновременно, благодаря системе диспетчеризации задач. Делятся основном на 3 вида:

1. Пользовательские;
2. Серверные или системы разделения времени — буквально, эффективно разделяют процессорное время между всеми запущенными процессами;
3. Встроенные находятся в основном в датчиках и микроконтроллерах, обслуживая физические процессы;

Дополнительно выделяют системы реального времени — время реагирования которых минимализируется (например, системы управления ПВО, которые должны быстро реагировать на вражеские ракеты), а также гипервизоры (Hypervisor) — класс операционных систем регулирующих работу других ОС (VirtualBox, QEMU).

Операционная система UNIX Операционная система UNIX была разработана в компании Bell Labs в начале 70-х годов. Первая версия была написана Кеном Томпсоном (Ken Thompson) на ассемблере для мини-компьютера PDP-7. Затем появилась вторая версия для компьютера PDP-11, уже на языке C. Ее автором был Деннис Ритчи (Dennis Ritchie). В 1974 году Ритчи и Томпсон опубликовали очень важную работу о системе UNIX. За эту работу они были награждены престижной премией Тьюринга Ассоциации вычислительной техники. После публикации многие университеты попросили у Bell Labs копию UNIX. Поскольку материнская компания Bell Labs, AT&T, была в тот момент регулируемой монополией и ей не разрешалось участвовать в компьютерном бизнесе, университеты смогли приобрести операционную систему UNIX за небольшую плату.

Машины PDP-11 использовались практически во всех компьютерных научных отделах университетов, но операционные системы, которые пришли туда вместе с PDP-11, не нравились ни профессорам, ни студентам. Эту нишу быстро заполнила операционная система UNIX, которая была снабжена исходными текстами, поэтому желающие могли дорабатывать ее до бесконечности.

Одним из первых университетов, купивших UNIX, был Калифорнийский университет в Беркли. Поскольку в наличии имелись все исходные коды, в Беркли сумели существенно преобразовать UNIX. Среди изменений было перенесение этой системы на миникомпьютер VAX, создание виртуальной памяти со страничной организацией, расширение имен файлов с 14 до 255 символов, а также поддержка сетевого протокола TCP/IP, который сейчас широко используется в Интернете (во многом благодаря тому факту, что был задействован в системе Berkeley UNIX).

Пока в Беркли шла вся эта работа, компания AT&T самостоятельно продолжала совершенствовать UNIX, в результате в 1982 году появилась операционная система System III, а в 1984 — System V. В конце 80-х годов широко использовались две разные и совершенно несовместимые версии UNIX: Berckeley UNIX и System V. Такое положение вещей вместе с отсутствием стандартов на форматы программ в двоичном коде в значительной степени препятствовало коммерческому успеху UNIX. Поставщики программного обеспечения не могли писать программы для UNIX, поскольку не было никакой гарантии, что эти программы смогут работать на любой версии UNIX. После долгих споров организация по стандартам в институте IEEE выпустила стандарт **POSIX** (Portable Operating System Interface) — **интерфейс переносимых операционных систем**, известный также как стандарт IEEE P1003. Сфера использования сетей в большей степени относится к Berckeley UNIX, а не System V. Именно в Беркли было введено понятие **сокета** как конечной точки сетевого соединения. Моделью для этой концепции стали 4-проводные настенные розетки для телефонов. Процесс в системе UNIX может создать сокет, подключиться к нему и установить соединение с сокетом на удаленном компьютере. Через это соединение можно затем пересылать данные в обоих направлениях, обычно по протоколу TCP/IP. Поскольку сетевые технологии в системе UNIX применялись десятилетиями, значительное число серверов в Интернете используют именно UNIX.

Категория	Примеры
Управление файлами	открытие или создание (open), чтение (read), запись (write), закрытие (close), удаление (unlink) и блокировка файлов (fcntl)
Управление каталогами	каталоги по своей сути те же файлы, так что операции по манипулированию файлами подходят и для каталогов
Управление процессами	порождение (fork), завершение (exit), трассировка процессов (ptrace) и передача сигналов
Управление памятью	разделение общей памяти между процессами, защита страниц (subpage_ prot)
Получение и установка параметров	идентификация пользователя, группы, процесса; установка приоритетов
Даты и периоды времени	установка времени доступа к файлам; использование временных интервалов; рабочий профиль программы
Работа в сети	установка и выделение соединений; отправка и получение сообщений
Прочее	учет использования ресурсов; манипуляция дисковыми квотами; перезагрузка системы

Ядро UNIX Самым нижним уровнем ядра UNIX является уровень драйверов устройств, которые изолируют файловую систему от аппаратного обеспечения. Изначально каждый драйвер устройства писался без учета всех остальных и представлял собой независимую единицу, поэтому возникали дублирования, которые были практически устранены с предложения Денниса Ритчи путем ввода новой структуры **потока ввода-вывода** (stream). С помощью потоков можно установить двустороннее соединение между процессом и устройством и вставить между ними один или несколько модулей.

Файловая система, как часть ядра Над драйверами устройств находится файловая система. Она управляет файлами, каталогами, размещением дисковых блоков, защитой и выполняет многие другие функции. В системе файлов имеется так называемый **кэш блоков**, предназначенный для хранения недавно считанных с диска блоков на случай, если они понадобятся еще раз. Файловая система имеет иерархический характер, причем вершиной этой иерархии является корневой каталог, который создается при вмонтировании файловой системы (ext4, ext5, NTFS) в точки монтирования. Каждому файлу ФС сопоставляется структура **индексных дескрипторов** i-node по 64 байта, которая содержит информацию о том, кто владеет файлом, что разрешено делать с файлом, где найти данные и т.п., а также число (к слову, корневой каталог всегда имеет i-node = 2). Число выделенных i-node'ов в ФС определяет число файлов, которые можно создать. Частой практикой является невозможность создать файл, несмотря на то, что на дисковом накопителе есть свободное место (сказывается нехватка i-node'ов).

Еще одна часть ядра системы UNIX — механизм управления процессами. Он выполняет различные функции, в том числе поддерживает взаимодействие между процессами (InterProcess Communication, IPC) и их синхронизацию, что позволяет избежать состояния гонок. Код управления процессами также занимается планированием процессов на основе их приоритетов. Кроме того, он обрабатывает сигналы, которые представляют собой особую (асинхронную) форму программных прерываний. Наконец, он управляет памятью. Большинство систем UNIX поддерживают виртуальную память с подкачкой страниц по требованию, иногда с некоторыми дополнительными особенностями (например, несколько процессов могут совместно использовать общие области адресного пространства).

Изначально ОС задумывалась как компактная система, благодаря чему должна была обеспечиваться надежность, производительность и отказоустойчивость. Первые версии UNIX были полностью текстовыми и ориентировались на терминалы, которые могли отображать 24 или 25 строк по 80 ASCII-символов. Пользовательским процессом управляла программа, которая называлась **оболочкой** и предоставляла интерфейс командной строки. Не является частью ядра, также как и графические интерфейсы GUI.

Файловая система подробнее С файловой системой тесно связана систем каталогов. Каждый пользователь может иметь несколько каталогов, а каждый каталог может содержать файлы и вложенные каталоги. Система UNIX обычно конфигурируется с главным каталогом (**корневым каталогом**), который содержит вложенные каталоги bin (для часто используемых программ), dev (для специальных файлов устройств ввода-вывода), lib (для библиотек) и usr (для пользовательских каталогов). Чтобы назвать файл,

нужно указать его **путь из корневого каталога**. Путь содержит список всех каталогов от корневого каталога к файлу, для разделения каталогов используется прямой слэш.

В каждый момент времени каждая работающая программа имеет **текущий каталог**. Путь может быть связан с текущим каталогом. Путь в котором не прописан корневой каталог называется **относительным путем**. Пользователь может создать **жесткую ссылку** на чужой файл, использовав для этого системный вызов link. Не разрешается создавать жесткие ссылки на каталоги, чтобы предотратить циклы в системе каталогов.

4 Операционная система Unix — интерпретаторы, стандартные потоки ввода-вывода, фильтры

Каждому открытому, читаемому, записываему сопоставляется небольшое целое число — **дескриптор файла**, которые идентифицирует файл при последующих вызовах (к слову, дескрипторы 0, 1, 2 соответствуют стандартным потокам stdin, stdout и stderr соответственно). Многие программы UNIX получают входные данные из стандартного ввода и записывают входные данные в старндартный вывод, такие программы называются **фильтрами**.

5 Операционная система Unix — основные команды, права файлов и способы их задания

С каждым файлом (а также каталогов, напоминаю, что это тоже файл) связана битовая карта, которая сообщает, кому разрешен доступ к файлу Карта содержит три поля RWX (Read, Write, eXecute) — чтение, запись, выполнение). Первое из них контролирует разрешение на чтение, запись и выполнение файлов для их владельца, второе — для других пользователей из группы владельца, третье — для всех остальных пользователей. Включение пользователей в те или иные группы осуществляется системным администратором, которого обычно называют **привилегированным пользователем**.