Языки описания схем

(mk.cs.msu.ru → Лекционные курсы → Языки описания схем)

Блок 12

Функциональные блоки процессора

лектор:

Подымов Владислав Васильевич

e-mail:

valdus@yandex.ru

Осень 2017

Вступление

В этом блоке слайдов начинается описание **микроархитектуры** модельного процессора

Как и раньше, предлагаемое описание намного более просто, чем микроархитектура реальных процессоров, но демонстрирует некоторые их базовые особенности

АЛУ

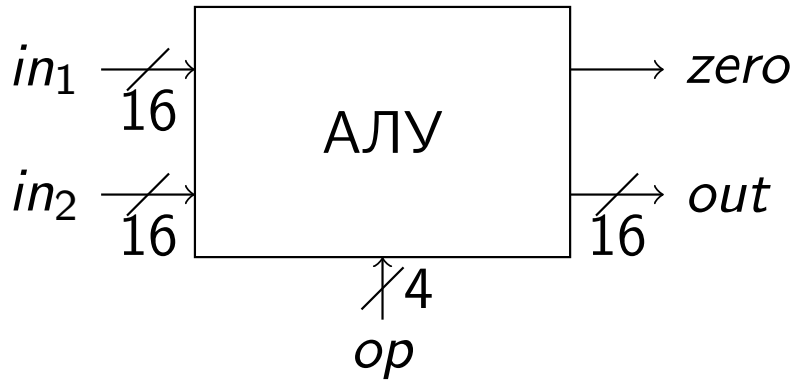
Многие операции требуют арифметико-логического преобразования значений:

- вычисление результата исполнения арифметико-логических команд
- проверка (не)равенства значений в регистрах при исполнении команд условного ветвления
- вычисление адреса ячейки в памяти данных при исполнении команд доступа к памяти
- ...

Арифметико-логические преобразования — ресурсоёмкий процесс, и создание отдельного блока для каждого типа преобразований довольно накладно

Все подобные преобразования принято выполнять в особом блоке (*модуле*) — арифметико-логическом устройстве (АЛУ)

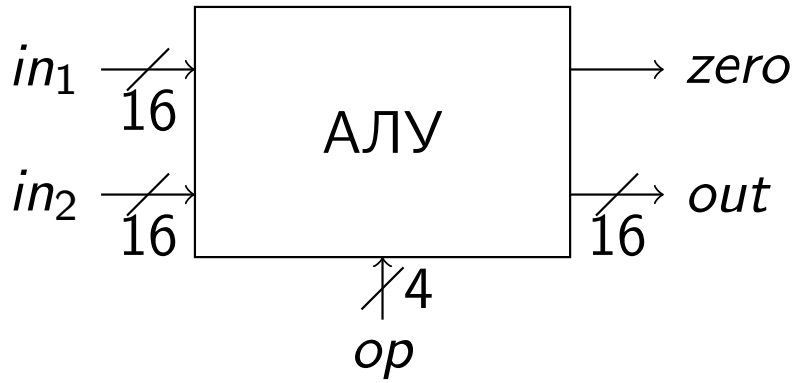
АЛУ



АЛУ — это комбинационный блок следующего вида:

- входы:
  - две шины ширины 16 (in_1, in_2), на которые подаются исходные числа
  - шина ширины 5 (op), на которую подаётся код применяемой арифметико-логической операции f
- выходы:
  - выход ширины 16 (out), на который непрерывно выдаётся значение $f(in_1, in_2)$
  - выходное логическое значение $zero$, непрерывно равное $1 \Leftrightarrow f(in_1, in_2) = 0$

АЛУ

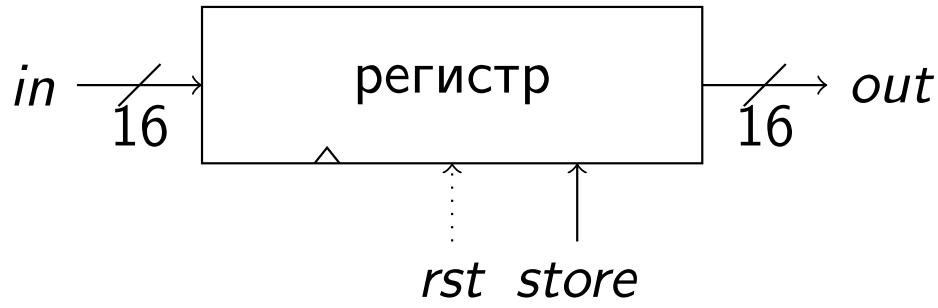


Выход *zero* будет использоваться при исполнении инструкций условного ветвления

В реальности АЛУ — намного более сложный блок:

- *оно может содержать другие входы и выходы, управляющие применением операции*
- *схема может быть и не комбинационной, если применяются “трудоёмкие” операции*
- *комбинационный функционал может быть настолько обширным, что критичной становится тонкая оптимизация схемы на многих уровнях представления*

Ячейка памяти

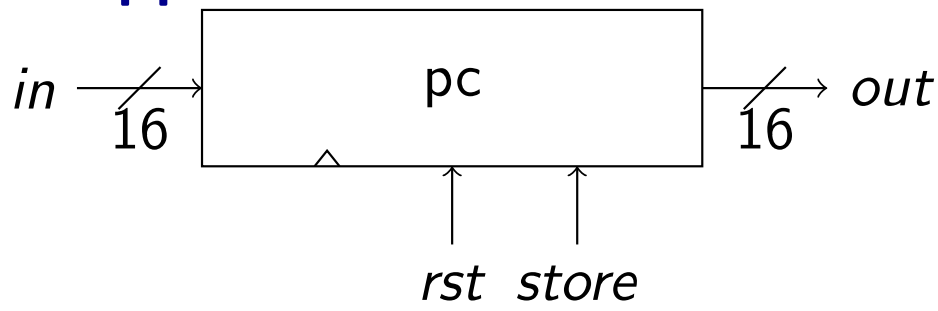


Во многих местах процессора и его окружения схожим образом хранятся значения заданной ширины (16), как со сбросом, так и без сброса:

- в счётчике команд
- в регистре
- в ячейке памяти данных

В этих местах в нашем модельном случае будет использоваться параллельный регистр с асинхронным включением загрузки: сохранение значения на входе происходит по переднему фронту тактового сигнала, и только если $store = 1$

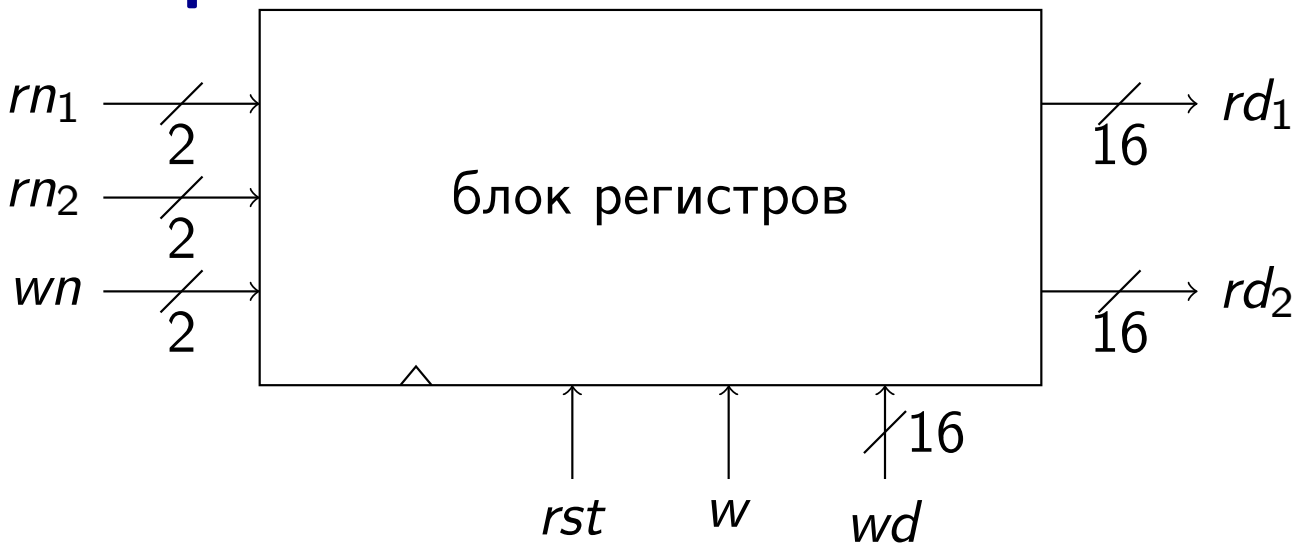
Счётчик команд



Счётчик команд — это просто ячейка памяти со сбросом

Управление счётчиком команд, как и другими функциональными блоками, вынесено за пределы этих блоков в другие специальные схемы

Блок регистров



В блоке регистров содержатся ячейки памяти всех четырёх регистров

На выходные шины rd_1, rd_2 непрерывно выдаются значения rn_1-го и rn_2-го регистров соответственно

Если $w = 1$, то по переднему фронту тактового сигнала в wn-й регистр записывается значение wd

По переднему фронту сигнала rst все хранимые значения перезаписываются нолями

Память

В реальном процессоре

- *внешняя память устроена довольно непросто:*
  - *она, как правило, связана с процессором через особую цифровую схему — контроллер*
  - *для доступа к ней используется особый коммуникационный протокол*
  - *устройство ячейки памяти зависит от целевого назначения этой памяти*
- *под памятью может пониматься многое, от оперативной и энергонезависимой памяти до “абстрактной” памяти для общения с периферией*
- *присутствуют специальные блоки организации работы с памятью, например:*
  - *блок виртуализации памяти, предоставляющий единообразный доступ ко всему, что может подразумеваться под “памятью”*
  - *блок кэш-памяти, ускоряющий выполнение команд доступа к памяти*

Память

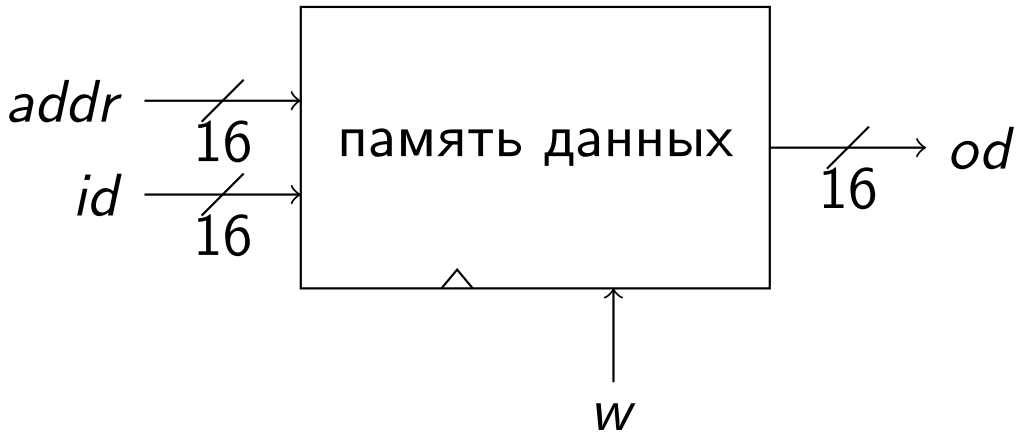
Для простоты будем рассматривать память,

- разделённую на память инструкций и память данных
  - *в противовес архитектуре фон Неймана, но ближе к реальным архитектурам*
- являющуюся составной частью процессора
  - *как кэш-память*
- содержащую выполняемую программу как часть дизайна
  - *как кэш-память инструкций, уже заполненная нужными ячейками*

Память инструкций

Память инструкций — это комбинационный блок, непрерывно выдающий в *com* инструкцию, **явно** и **неизменно** записанную в *addr*-й ячейке

Память данных



Память данных устроена примерно так же, как и блок регистров, но с немного другим набором входов и выходов

На выход *od* непрерывно выдаётся значение, записанное в *addr*-й ячейке

Если $w = 1$, то по переднему фронту тактового сигнала в *addr*-ю ячейку записывается значение *id*

Конец блока 12