

ПРОЦЕССОР LC-3

1. Введение	3
2. Информация о процессоре	3
2.1. Система команд	3
2.2. Обзор LC-3: Память и регистры	4
2.3. Обзор LC-3: Набор инструкций	4
3. Флаги состояния	4
4. Режимы адресации	5
4.1. Режим адресации относительно PC	5
4.2. Косвенный режим адресации	5
4.3. Режим адресации база+смещение	6
4.4. Адресация с загрузкой оптимального адреса	6
5. Команды	6
5.1. Арифметические команды	6
5.1.1. Общая информация о арифметических командах	6
5.1.2. NOT	7
5.1.3. ADD/AND (Регистровый режим)	8
5.1.4. ADD/AND (Непосредственный режим)	9
5.1.5. Реализация остальных арифметических операций	9
5.2. Команды по перемещению данных	10
5.2.1. Общая информация о командах по перемещению данных	10
5.2.2. LD (смещение относительно PC)	11
5.2.3. LDI (косвенная адресация)	12
5.2.4. LDR (адресация база+смещение)	13
5.2.5. ST (смещение относительно PC)	14
5.2.6. STI (косвенная адресация)	15
5.2.7. STR (адресация база+смещение)	16
5.2.8. LEA (Прямая загрузка)	17
5.3. Управляющие команды	18
5.3.1. BR (условный переход)	19
5.3.2. JMP (безусловный переход)	21
5.3.3. TRAP (прерывания)	22
6. Функциональные элементы	23
6.1. Блоксхема процессора	23
6.2. Общая шина	23
6.3. Память	23
6.4. АЛУ (Арифметико-логическое устройство)	23
6.5. Регистровый файл	23
6.6. Счётчик команд (PC) и Мультиплексор счётчика команд (PCMUX)	23
6.7. Адресный регистр памяти (MAR) и Мультиплексор адресного регистра памяти (MARMUX)	24

6.8. Логика признаков результата	24
6.9. Управляющий блок — конечный автомат (Control Unit – Finite State Machine)	24

ВВЕДЕНИЕ

Я решил написать этот конспект, являющийся переводом презентации Техасского университета в Остине, для того, чтобы разобраться в процессоре LC-3 и для дальнейшего использования конспекта при создании его эмулятора. Я не нашел подробной инструкции к этому процессору на русском языке, так что это первые подобные методические указания.

Оригинальный источник:

https://www.cs.utexas.edu/~fussell/courses/cs310h/lectures/Lecture_10-310h.pdf

Ссылка на репозиторий:

<https://github.com/vodobryshkin/LC-3>

ИНФОРМАЦИЯ О ПРОЦЕССОРЕ

СИСТЕМА КОМАНД

Что такое **система команд процессора (СК)** (Instruction Set Architecture)?

СК = Все видимые программисту компоненты и операции компьютера.

Это в том числе:

- Организация памяти:
 - адресное пространство – как обращаться к ячейкам памяти?
 - адресуемость – сколько бит на ячейку памяти?
- Набор регистров:
 - Сколько регистров?
 - Какого они размера?
 - Как они могут использоваться?
- Набор инструкций:
 - Коды операций (Opcodes)
 - Типы данных
 - Режимы адресации

СК даёт всю информацию необходимую для программиста, который хочет написать программу на машинном коде.

ОБЗОР LC-3: ПАМЯТЬ И РЕГИСТРЫ

- Память
 - Адресное пространство составляет 2^{16} ячеек памяти
 - Все адреса 16-битные
- Регистры

Временное хранилище данных, доступ за один машинный цикл. Обычно доступ к памяти занимает больше чем один машинный цикл.

 - Восемь 16-битных регистров общего назначения: **R0-R7**
 - Другие регистры не являются адресуемыми напрямую, но используются разными командами. Самыми важными являются **PC** (Program counter – счётчик команд) и **флаги результата**.

ОБЗОР LC-3: НАБОР ИНСТРУКЦИЙ

- Коды команд
 - 15 команд
 - **Арифметические команды:** ADD, AND, NOT
 - **Команды по перемещению данных:** LD, LDI, LDR, LEA, ST, STR, STI
 - **Управляющие команды:** BR, JSR/JSRR, JMP, RTI, TRAP

Некоторые команды устанавливают/сбрасывают флаги результата по прошествию операции (N = результат отрицательный, Z = результат равен нулю, P = результат положительный)

- Тип данных:

16-битные числа в дополнительном коде.
- Режимы адресации:

Как определяется местоположение операнда? Существуют такие режимы адресации:

 - Не использующие память: непосредственный, регистровый
 - Использующие память: относительно PC, косвенный, базовый со смещением.

ФЛАГИ СОСТОЯНИЯ

LC-3 имеет три флага состояния (отдельные регистры):

- N – negative (меньше нуля)
- Z – zero (число равно нулю)
- P – positive (больше нуля)

Флаг выставляется после любой команды, которая записывает данные в регистр (то есть команды ADD, AND, NOT, LD, LDR, LDI, LEA)

Один и только один флаг будет выставлен постоянно.

РЕЖИМЫ АДРЕСАЦИИ

РЕЖИМ АДРЕСАЦИИ ОТНОСИТЕЛЬНО РС

Проблема: есть необходимость указать адрес напрямую в инструкции. Однако адрес 16-битный, как и команда. После выделения 4 бит на код операции и 3 бит под регистр, остаётся всего 9 бит для адреса. как же быть?

Решение:

Использовать 9 бит как *знаковое смещение* от текущего значения в РС.

Так как под смещение выделено 9 бит, то справедливо следующее выражение:

$$-256 \leq \text{смещение} \leq 255$$

Тогда, можно обратиться к любому адресу X , такому что:

$$PC - 256 \leq X \leq PC + 255$$

Стоит помнить, что РС увеличивается на 1 во время цикла выборки команды (FETCH) и это происходит до этапа вычисления адреса.

КОСВЕННЫЙ РЕЖИМ АДРЕСАЦИИ

Проблема: используя режим адресации относительно РС, можно получать доступ к данным только в пределах 256 ячеек. Но как получить значения из остальной памяти?

Решение:

Прочитать адрес из другой ячейки памяти и затем выполнить загрузку/сохранение из/в нужный адрес.

Первый адрес вычисляется как в режиме адресации относительно РС, затем значение из этой ячейки используется как фактический адрес для операции загрузки/сохранения

РЕЖИМ АДРЕСАЦИИ БАЗА+СМЕЩЕНИЕ

Проблема: используя режим адресации относительно РС, можно получать доступ к данным только в пределах 256 ячеек. Но как получить значения из остальной памяти?

Вычисляет адрес аналогично РС-относительной адресации (РС плюс знаковое смещение) и сохраняет результат вычисления (полученный адрес) в указанный регистр

Решение:

Использовать регистр для генерации полного 16-битного адреса.

4 бита выделяется подкод команды, 3 для Src/Dest регистра, 3 бита для базового регистра и остальные 6 бит для базового смещения.

Смещение знаково расширяется перед добавлением к базовому регистру.

АДРЕСАЦИЯ С ЗАГРУЗКОЙ ОПТИМАЛЬНОГО АДРЕСА

Напрямую загружается смещение в регистр.

КОМАНДЫ

АРИФМЕТИЧЕСКИЕ КОМАНДЫ


ОБЩАЯ ИНФОРМАЦИЯ О АРИФМЕТИЧЕСКИХ КОМАНДАХ

В LC-3 всего три арифметические операции: ADD, AND, NOT.

- Операндами являются регистры (над регистром производится операция и результат записывается в другой регистр).
- Инструкции не затрагивают память.
- ADD и AND могут использовать “непосредственный” режим, когда один операнд жестко связан с инструкцией.

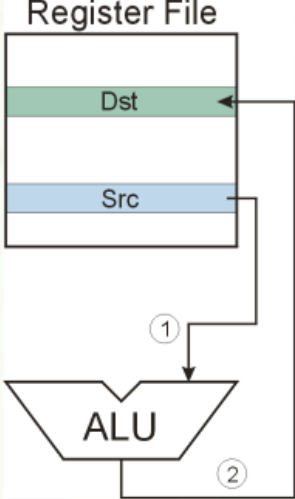
Далее будет описано, когда и куда перемещаются данные для желаемой операции.

NOT

 NOT (Register)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT	1	0	0	1	Dst			Src			1	1	1	1	1	1

Register File



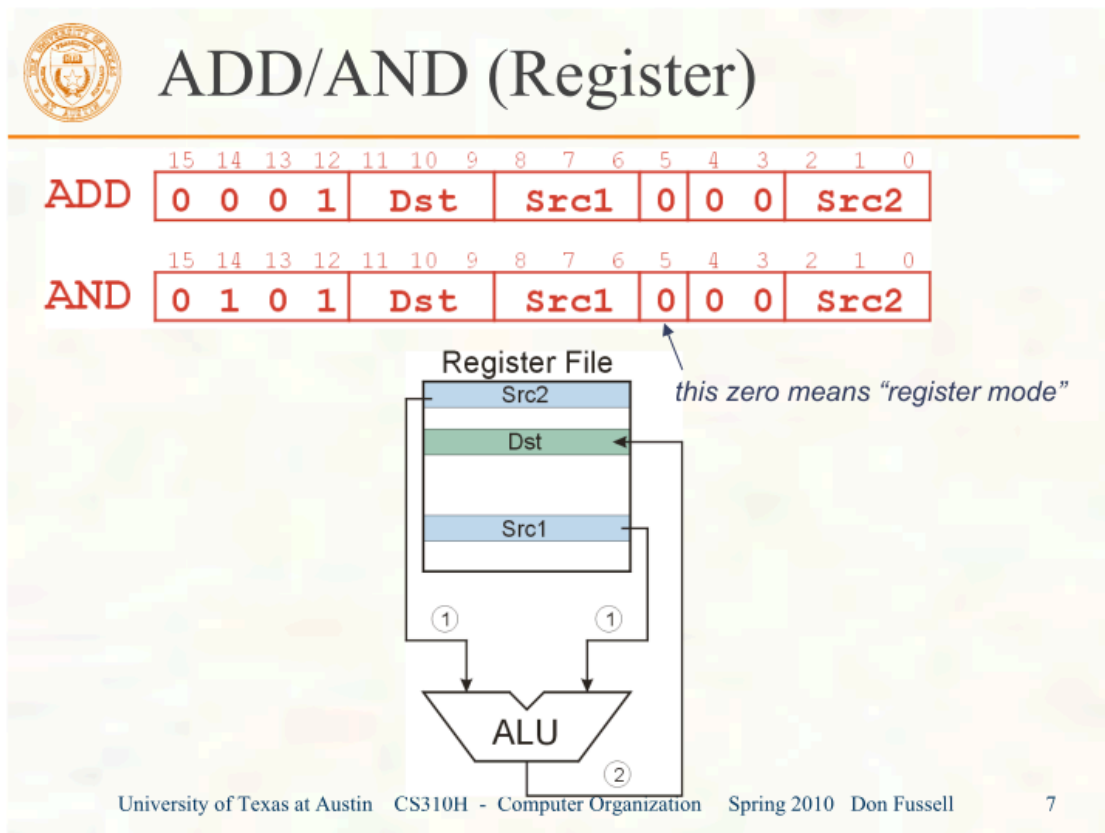
Note: Src and Dst could be the same register.

University of Texas at Austin CS310H - Computer Organization Spring 2010 Don Fussell 6

- Код команды:
1001 Dst Src 111111
- Цикл исполнения команды:
 1. Корневой регистр подаётся на правый вход АЛУ
 2. Значение обрабатывается в АЛУ
 3. Результат записывается на результирующий регистр

В качестве операнда, над которым производят действия, и результирующего операнда может выступать один регистр.

ADD/AND (РЕГИСТРОВЫЙ РЕЖИМ)

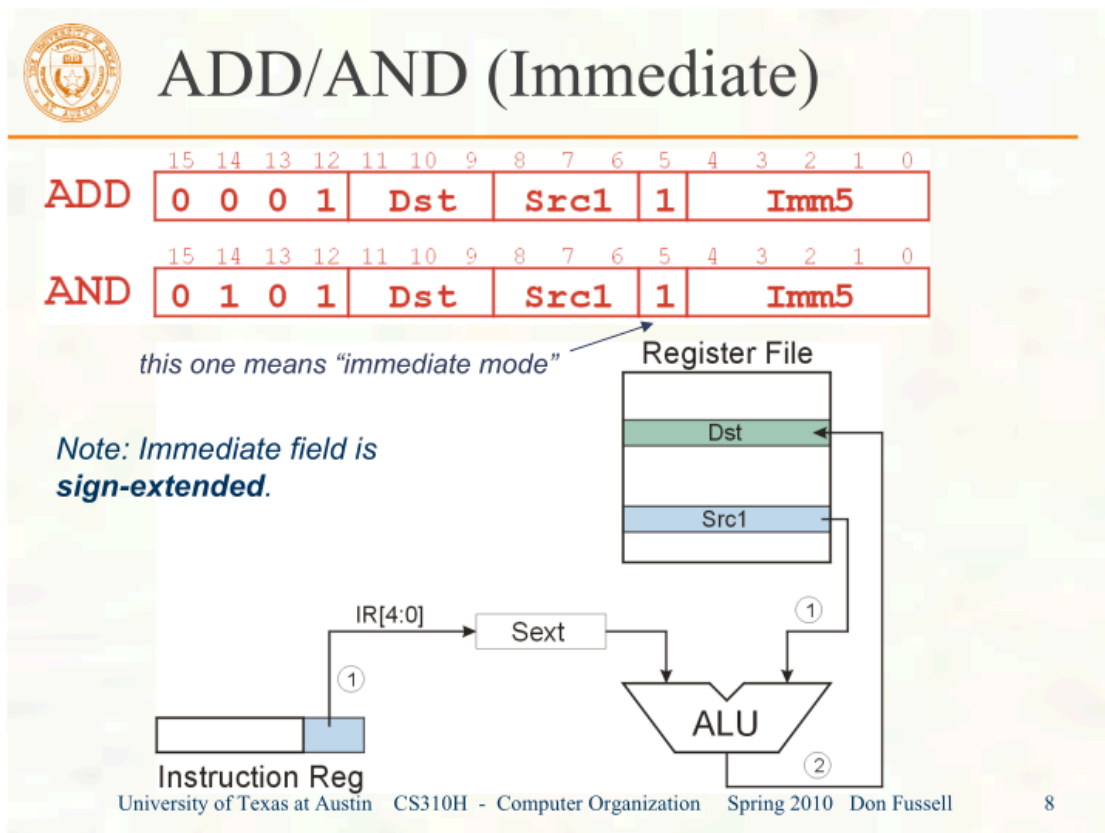


- Код команд:
ADD – 0001 Dst Src1 0 00 Src2
AND – 0101 Dst Src1 0 00 Src2

Ноль в 5 бите (нумерация с 0) означает регистровый режим операции.

- Цикл исполнения команды:
 1. Значение из первого регистра подаётся на левый вход АЛУ
 2. Значение из второго регистра подаётся на правый вход АЛУ
 3. Значения обрабатываются в АЛУ
 4. Результат записывается на результирующий регистр

ADD/AND (НЕПОСРЕДСТВЕННЫЙ РЕЖИМ)



- Код команд:
ADD – 0001 Dst Src1 1 Imm5
AND – 0101 Dst Src1 1 Imm5

Единица в 5 бите (нумерация с 0) означает непосредственный режим операции.

- Цикл исполнения команды:
 1. Значение из корневого регистра подаётся на правый вход АЛУ
 2. У значения Imm5 расширяется знак и подаётся на левый вход АЛУ
 3. Значения обрабатываются в АЛУ
 4. Результат записывается на результирующий регистр

РЕАЛИЗАЦИЯ ОСТАЛЬНЫХ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ

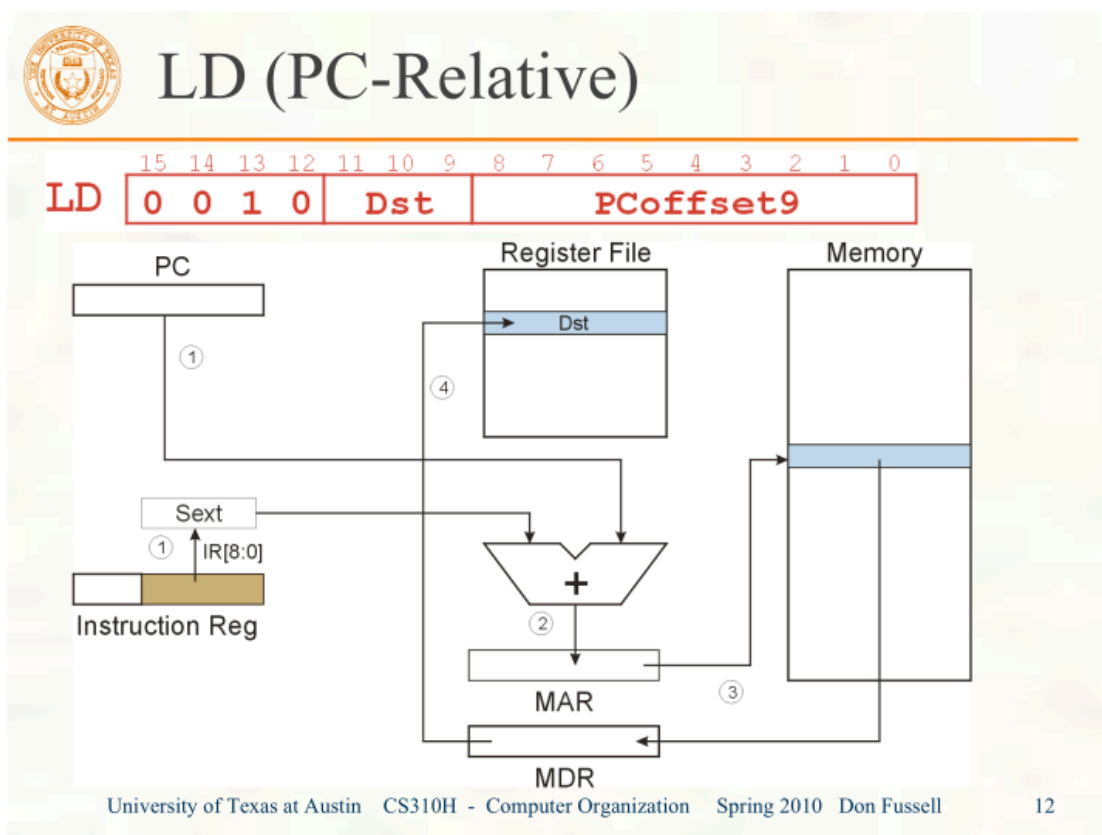
Все другие арифметические и логические операции могут быть написаны программистом вручную, используя данные базовые инструкции.

КОМАНДЫ ПО ПЕРЕМЕЩЕНИЮ ДАННЫХ

ОБЩАЯ ИНФОРМАЦИЯ О КОМАНДАХ ПО ПЕРЕМЕЩЕНИЮ ДАННЫХ

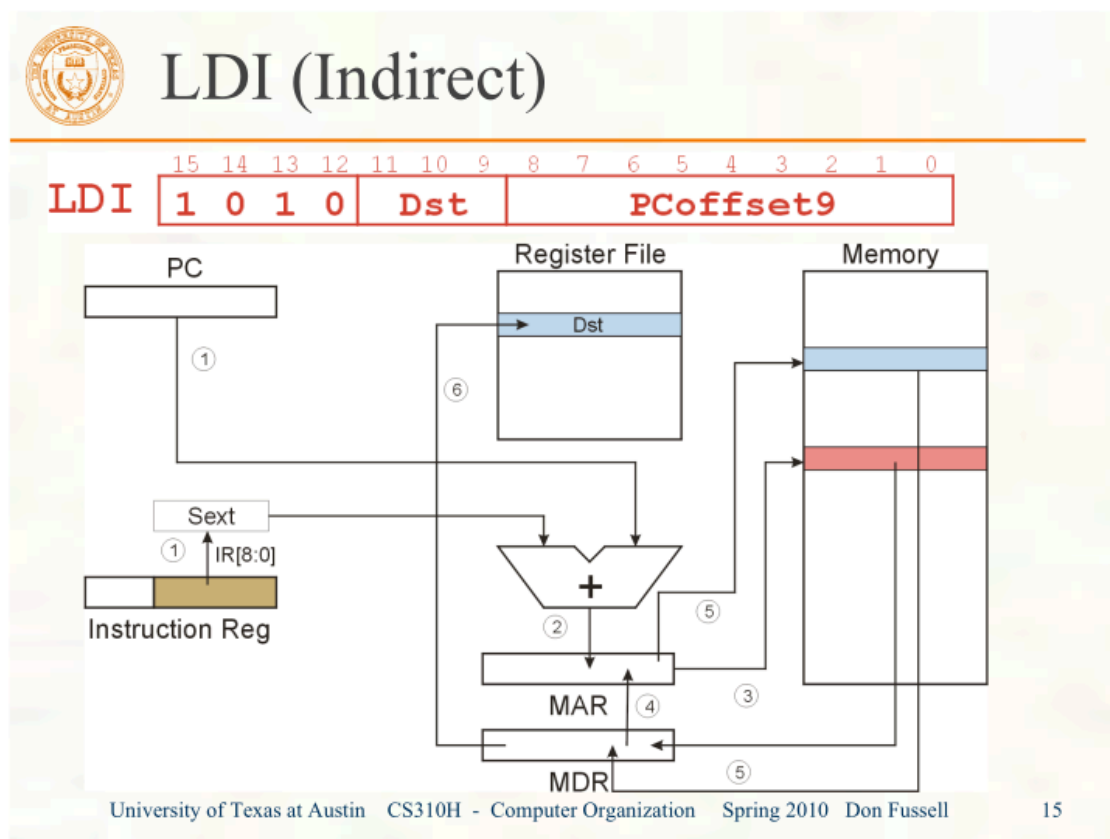
- Загрузка (Load) – чтение данных из памяти в регистр
 - LD: адресация относительно PC
 - LDR: базовая адресация со смещением
 - LDI: косвенная адресация
- Сохранение (Store) – запись данных из регистра в память
 - ST: адресация относительно PC
 - STR: базовая адресация со смещением
 - STI: косвенная адресация
- Загрузка эффективного адреса (LEA) — вычисление адреса и сохранение в регистр
 - LEA: непосредственная адресация
 - Не обращается к памяти

LD (СМЕЩЕНИЕ ОТНОСИТЕЛЬНО PC)



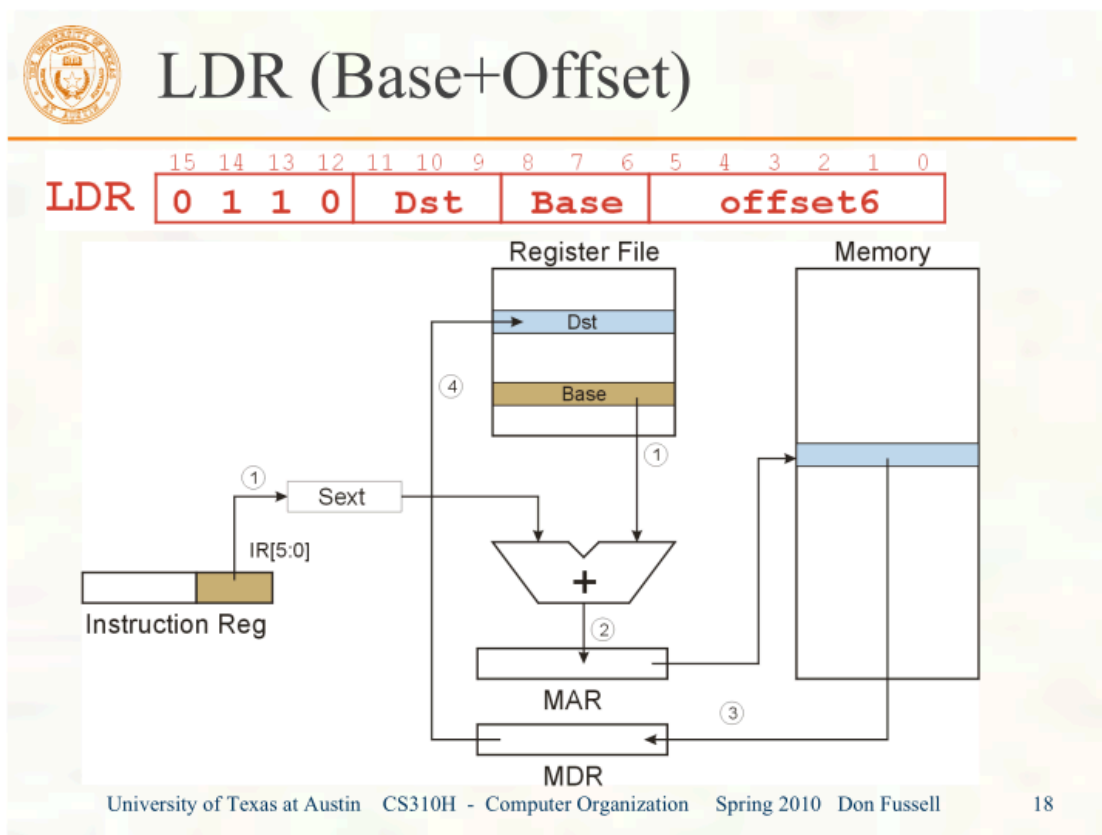
- Код команды:
0010 Dst PCOffset9
- Цикл исполнения команды:
 1. Значение PC подаётся на правый вход АЛУ
 2. 9 бит из Instruction Register (PCOffset9) расширяют знак и подаются на левый вход АЛУ
 3. В АЛУ PC складывается с расширенным значением PCOffset9 и записывается в MAR (Memory Address Register)
 4. По адресу из MAR происходит чтение содержимого памяти и запись в MDR – Memory Data Register
 5. Результат записывается в регистр Dst

LDI (КОСВЕННАЯ АДРЕСАЦИЯ)



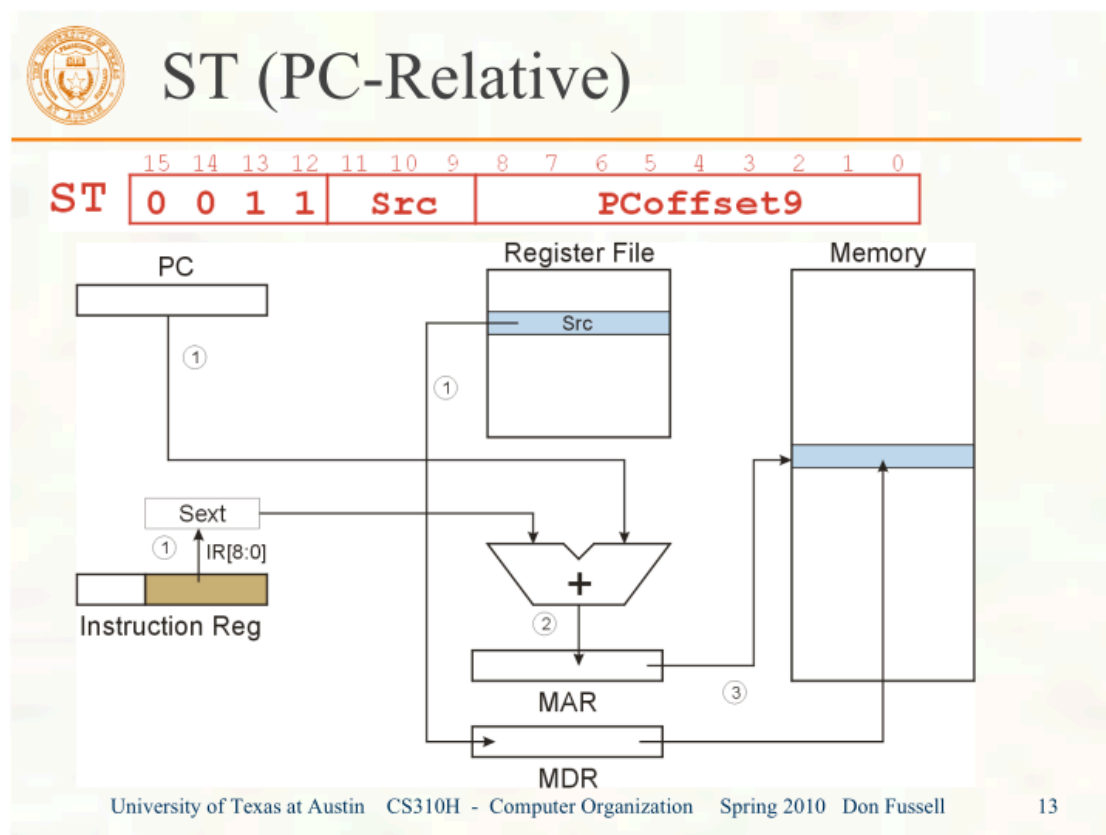
- Код команды:
1010 Dst PCoffset9
- Цикл исполнения команды:
 1. Значение PC подаётся на правый вход АЛУ
 2. 9 бит из Instruction Register (PCoffset9) расширяют знак и подаются на левый вход АЛУ
 3. В АЛУ PC складывается с расширенным значением PCoffset9 и записывается в MAR
 4. По адресу из MAR происходит чтение содержимого первой ячейки памяти (с фактическим адресом) и запись в MDR
 5. Значение MDR записывается в MAR
 6. По адресу из MAR происходит чтение содержимого памяти и запись в MDR
 7. Результат записывается в регистр Dst

LDR (АДРЕСАЦИЯ БАЗА+СМЕЩЕНИЕ)



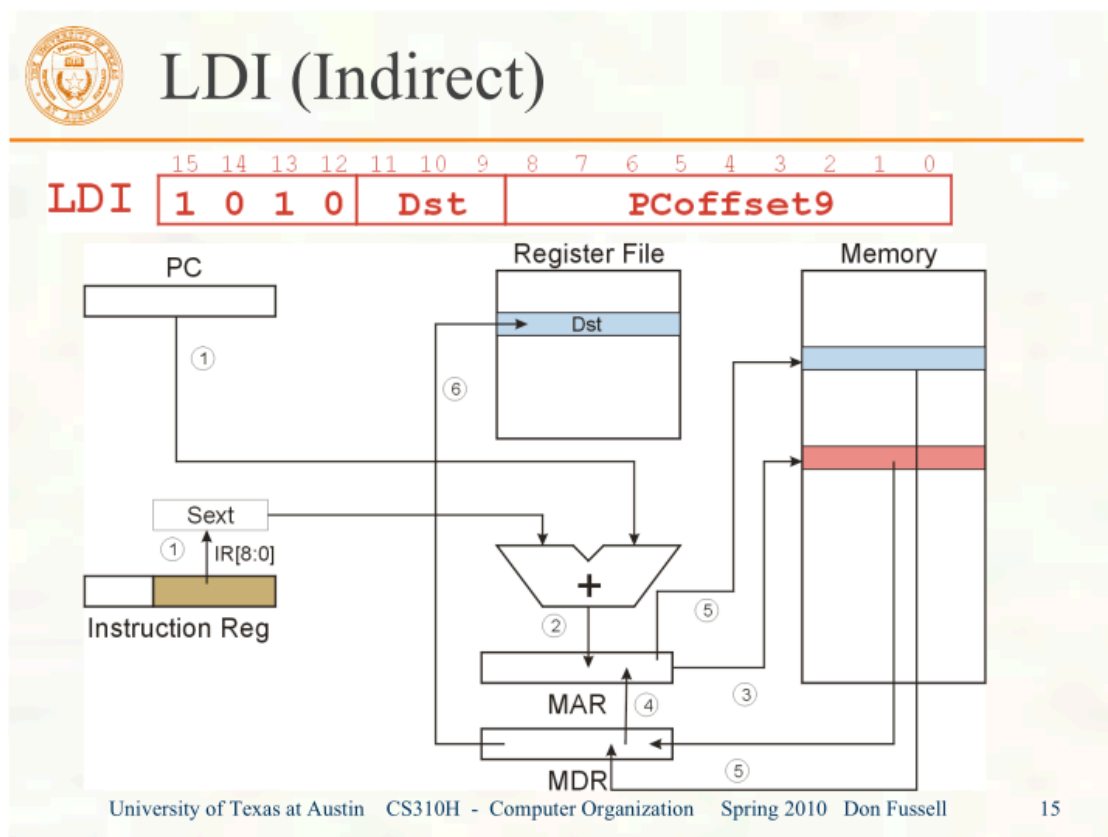
- Код команды:
0110 Dst Base offset6
- Цикл исполнения команды:
 1. Значение из регистра Base подаётся на правый вход АЛУ
 2. 6 бит из IR (offset6) расширяют знак и подаются на левый вход АЛУ
 3. В АЛУ Base складывается с расширенным значением offset6 и записывается в MAR (Memory Address Register)
 4. По адресу из MAR происходит чтение содержимого памяти и запись в MDR
 5. Результат записывается в регистр Dst

ST (СМЕЩЕНИЕ ОТНОСИТЕЛЬНО PC)



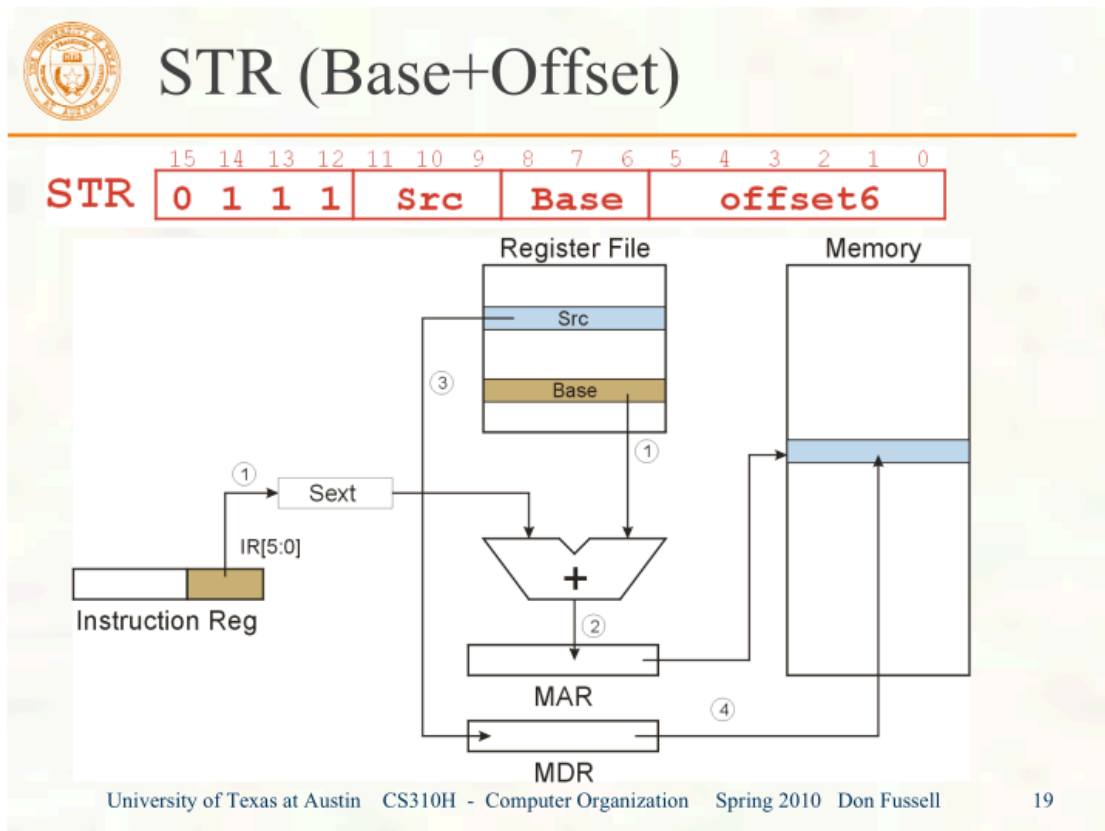
- Код команды:
0011 Dst PCoffset9
- Цикл исполнения команды:
 1. Значение регистра Src записывается в регистр MDR
 2. Значение PC подаётся на правый вход АЛУ
 3. 9 бит из Instruction Register (PCoffset9) расширяют знак и подаётся на левый вход АЛУ
 4. В АЛУ PC складывается с расширенным значением PCoffset9 и записывается в MAR (Memory Address Register)
 5. По адресу из MAR происходит запись содержимого MDR

STI (КОСВЕННАЯ АДРЕСАЦИЯ)



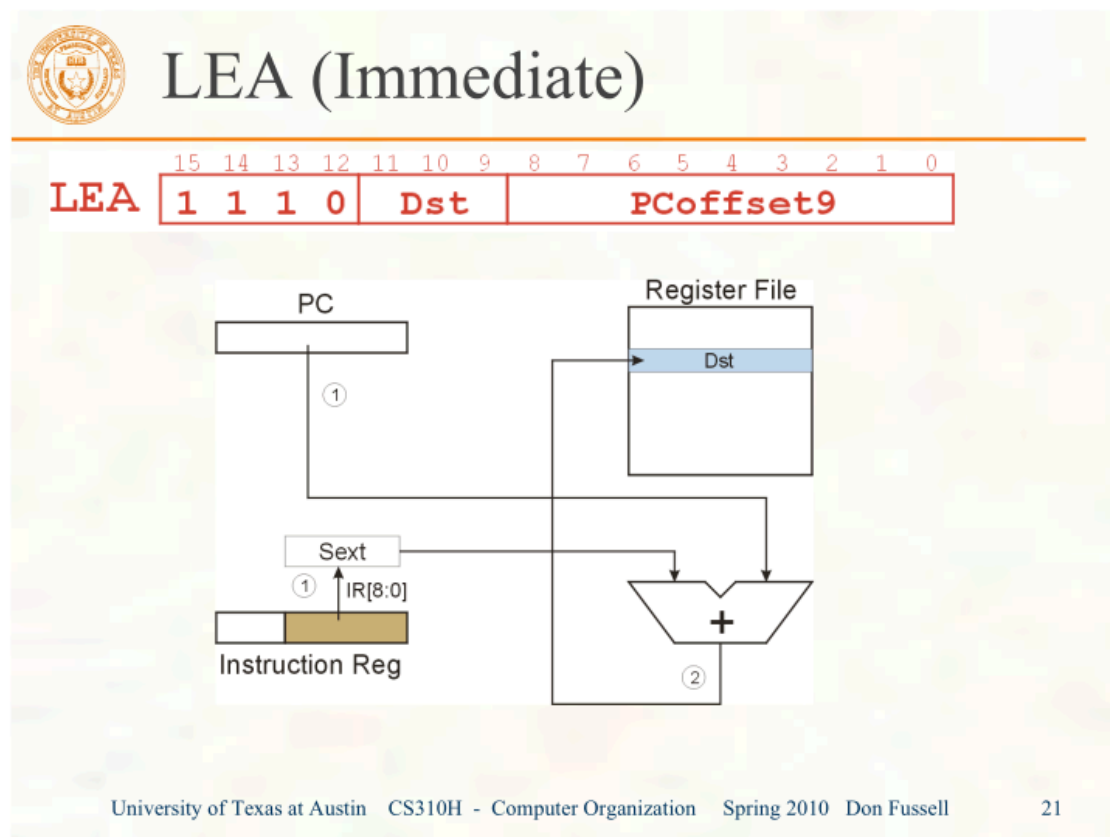
- Код команды:
1011 Dst PCOffset9
- Цикл исполнения команды:
 1. Значение PC подаётся на правый вход АЛУ
 2. 9 бит из Instruction Register (PCOffset9) расширяют знак и подаются на левый вход АЛУ
 3. В АЛУ PC складывается с расширенным значением PCOffset9 и записывается в MAR
 4. По адресу из MAR происходит чтение содержимого первой ячейки памяти (с фактическим адресом) и запись в MDR
 5. Значение MDR записывается в MAR
 6. Значение из регистра Src записывается в MDR
 7. По адресу из MAR происходит запись содержимого MDR

STR (АДРЕСАЦИЯ БАЗА+СМЕЩЕНИЕ)



- Код команды:
0111 Src Base offset6
- Цикл исполнения команды:
 1. Значение Base подаётся на правый вход АЛУ
 2. 6 бит из Instruction Register (offset6) расширяют знак и подаются на левый вход АЛУ
 3. В АЛУ Base складывается с расширенным значением offset6 и записывается в MAR
 4. Значение регистра Src записывается в регистр MDR
 5. По адресу из MAR происходит запись содержимого MDR

LEA (ПРЯМАЯ ЗАГРУЗКА)



- Код команды:
1110 Dst Base PCoffset9
- Цикл исполнения команды:
 1. Значение PC подаётся на правый вход АЛУ
 2. 9 бит из Instruction Register (PCoffset9) расширяют знак и подаются на левый вход АЛУ
 3. В АЛУ PC складывается с расширенным значением PCoffset9 и записывается в Dst

УПРАВЛЯЮЩИЕ КОМАНДЫ

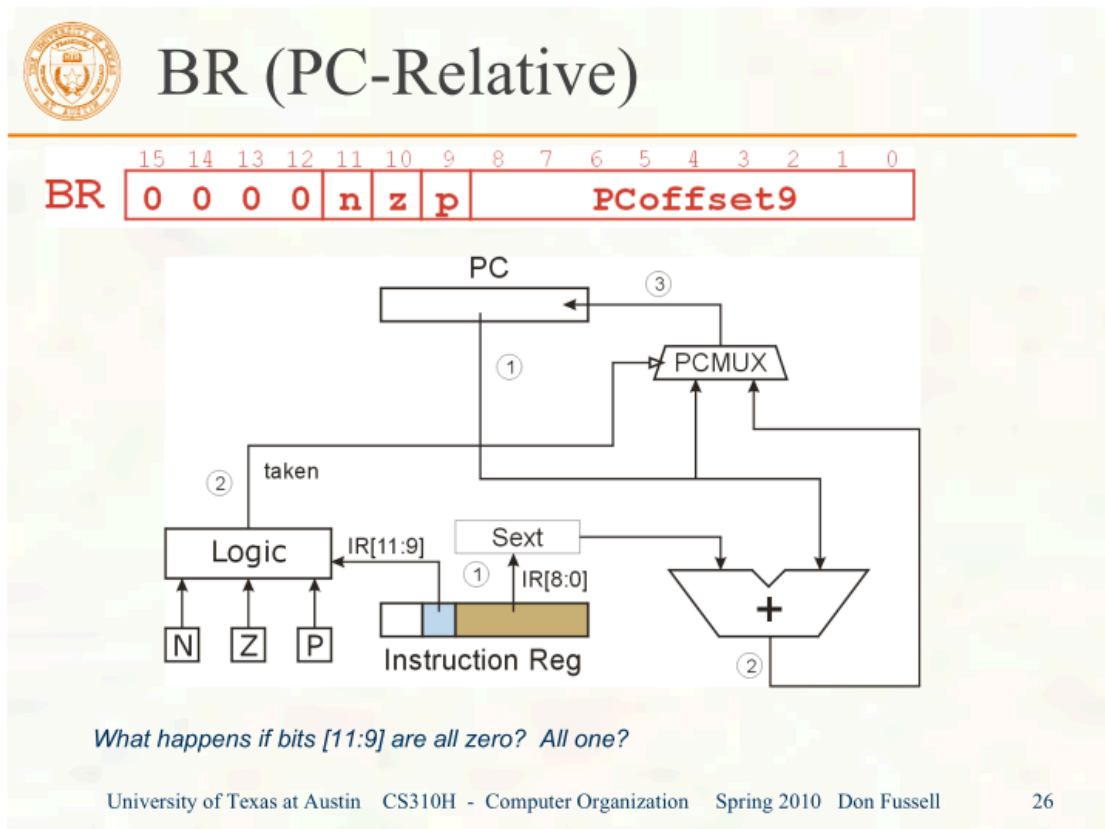
Используются для изменения последовательности выполнения команд (благодаря изменению РС). Управляющие команды можно разбить:

- Условные переходы
 - Переход совершается, если указанное условие выполняется (указанное смещение добавляется к РС для формирования нового РС)
 - Иначе перехода не происходит (РС не меняется, переходит к следующей инструкции по инкременту как и должно быть)
- Безусловные переходы (команда Jump) – всегда меняет РС
- Прерывание
 - Изменяет РС на адрес одного из векторов прерываний
 - После выполнения процедура вернёт управление следующей инструкции (после TRAP)

BR (УСЛОВНЫЙ ПЕРЕХОД)

Команда условного перехода использует один или более флагов состояния.

- Если нужный бит установлен, то переход происходит.
- Если перехода не происходит, то PC изменяется на адрес следующей инструкции в памяти.




- Код команды:
0000 n z p PCoffset9
- Цикл исполнения команды:
 1. Значения флагов N, Z, P считываются из регистра состояния и подаются на вход логического блока
 2. 3 бита из Instruction Register (IR[11:9]) подаются на логический блок для сравнения с флагами
 3. Если условие ветвления (taken) истинно, то 9 бит из Instruction Register (PCoffset9) расширяются по знаку (Sext)
 4. Расширенное значение PCoffset9 подаётся на левый вход АЛУ, а текущее значение PC — на правый вход АЛУ
 5. АЛУ выполняет сложение PC и расширенного PCoffset9, результат подаётся на мультиплексор PCMUX

6. Если $\text{taken} = 1$, мультиплексор выбирает результат сложения и записывает его в РС, иначе РС остаётся без изменений

JMP (БЕЗУСЛОВНЫЙ ПЕРЕХОД)

Команда условного перехода использует одну или более флагов состояния.

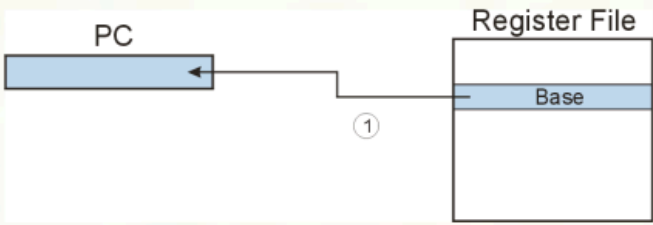
- Переход происходит в любом случае.



JMP (Register)

- Jump is an unconditional branch -- *always* taken.
- Target address is the contents of a register.
- Allows any target address.


	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JMP	1	1	0	0	0	0	0	Base			0	0	0	0	0	0



University of Texas at Austin CS310H - Computer Organization Spring 2010 Don Fussell 29

- Код команды:
1100 000 Base 000000
- Цикл исполнения команды:
 1. Значение Base переносится в PC

TRAP (ПРЕРЫВАНИЯ)



TRAP

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRAP	1	1	1	1	0	0	0	0	trapvect8							

- Calls a **service routine**, identified by 8-bit “trap vector.”

vector	routine
x23	input a character from the keyboard
x21	output a character to the monitor
x25	halt the program

- When routine is done,
PC is set to the instruction following TRAP.
- (We’ll talk about how this works later.)

University of Texas at Austin CS310H - Computer Organization Spring 2010 Don Fussell 30

Делает один из трёх системных вызовов, по id конкретного вектора прерываний.

Вектор
Системный вызов
x23
Ввод символа с клавиатуры
x21
Вывод символа на монитор
x25
Завершение работы программы

Когда вызов закончен, возвращает PC на инструкцию, следующую после прерывания.

ФУНКЦИОНАЛЬНЫЕ ЭЛЕМЕНТЫ

БЛОКСХЕМА ПРОЦЕССОРА

ОБЩАЯ ШИНА

Общая шина – специальный набор проводов, передающий 16-битный сигнал ко многим компонентам.

Входы шины — это “трехстабильные устройства”, которые подают сигнал на шину только при активации

1. В любой момент времени должен быть активен только один (16-битный) сигнал
2. Управляющий блок решает, какой сигнал “управляет” шиной
3. Любое количество компонентов может читать данные с шины
4. Регистр фиксирует данные с шины только при активации записи управляющим блоком

ПАМЯТЬ

Память состоит из управляющих регистров и регистров данных для памяти и устройств ввода-вывода: MAR (регистр адреса памяти), MDR (регистр данных памяти) (а также управляющие сигналы для чтения/записи).

АЛУ (АРИФМЕТИКО-ЛОГИЧЕСКОЕ УСТРОЙСТВО)

АЛУ принимает ввод из регистрового файла и из знакорасширенных битов из IR (в прямом режиме), вывод идёт на общую шину.

РЕГИСТРОВЫЙ ФАЙЛ

1. Обладает двумя адресами для чтения (SR1, SR2), одним адресом для записи (DR). Входные данные поступают с шины, есть два 16битных входа.

СЧЁТЧИК КОМАНД (PC) И МУЛЬТИПЛЕКСОР СЧЁТЧИКА КОМАНД (PCMUX)

В счётчик команд могут поступить три вида значения, контролируемые **PCMUX**:

1. Цикл выборки операнда (FETCH) – PC + 1

2. Добавление адреса (при BR и JMP)
3. Шина прерываний (TRAP)

АДРЕСНЫЙ РЕГИСТР ПАМЯТИ (MAR) И МУЛЬТИПЛЕКСОР АДРЕСНОГО РЕГИСТРА ПАМЯТИ (MARMUX)

В **MAR** может поступить два типа значений, контролируемые **MARMUX**:

1. Добавление адреса при командах LD/ST, LDR/STR
2. Знаково расширенное значение из IR[7:0] – при прерываниях TRAP

ЛОГИКА ПРИЗНАКОВ РЕЗУЛЬТАТА

По значению на шине генерирует N, Z, P флаги.

Регистры устанавливаются только при активации управляющим блоком (LD.CC) (при командах ADD, AND, NOT, LD, LDI, LDR, LEA)

УПРАВЛЯЮЩИЙ БЛОК — КОНЕЧНЫЙ АВТОМАТ (CONTROL UNIT – FINITE STATE MACHINE)

На каждом машинном цикле изменяет управляющие сигналы для следующей фазы обработки инструкции:

1. Какой компонент управляет шиной? (GatePC, GateALU, ...)
2. Какие регистры разрешены для записи? (LD.IR, LD.REG, ...)
3. Какую операцию должно выполнить АЛУ? (ALUK)
4. и др.

Включает декодер для кода операции (opcode) и др. логику.