# 1. Структурная схема RISC процессора i80860

Структурная схема RISC процессора i80860 представлена на рисунке 1 и вкладке “Процессор” программы моделирования. Процессор состоит из следующих основных элементов: **пятиступенчатый конвейер команд**, содержащий исполняемые в данный момент команды, находящиеся на разных стадиях готовности;
**кэш команд**
объемом 4 Кб и
**кэш данных**
объемом 8 Кб; счетчик команд
**СК**
и счетчик тактов
**СТ**
; регистр адреса
**РА**
, регистр чтения
**РЧ**
и трехступенчатый конвейерный регистр записи
**РЗ**
, предназначенные для обмена с кэшем данных;
**блок регистров общего назначения**
(РОНов); фиксаторы операндов
**Ф1**
и
**Ф2**
и фиксатор результата операции
**ФД**
, предназначенные для временного хранения операндов и результата выполненной операции;
**коммутатор**
первого операнда, предназначенный для выбора Ф1 в качестве первого операнда либо счетчик
**СК**
; флажок условия
**С**
, используемый при условных переходах;
**арифметико логическое устройство**
(АЛУ) и
**сдвигатель**
.

# 2.

# Описание функциональных узлов процессора

Конвейер команд процессора состоит из пяти ступеней. Типичные команды выполняются на четырех первых ступенях и проходят четыре стадии: выборка команды из кэша (**F**etchi ng), декодирование и выборка операндов (
**D**
ecoding), исполнение (
**E**
xecuting) и запись результата (
**W**
riting). Команда чтения из памяти исполняется на всех пяти стадиях конвейера: выборка команды из кэша (
**F**
etching), декодирование и выборка операндов (
**D**
ecoding), исполнение (
**E**
xecuting), чтение из памяти (
**R**
eading) и запись прочитанного значения в регистр (
**W**
riting). На каждом тактовом цикле происходит продвижение команд по ступеням конвейера.

Кэш команд представляет собой память, организованную в виде 1024 двойных слов (32 разряда). Адресация команд осуществляется счетчиком команд **CK**.

Кэш данных содержит 8192 байт. Обмен с кэшем может осуществляться байтами, словами или двойными словами. Адресация данных осуществляется регистром адреса **РА**.
Прочитанные данные помещаются в регистр чтения **РЧ**. Записываемые данные на стадии декодирования помещаются на первую ступень конвейерного регистра записи **РЗ**.

В лабораторной модели кэши данных и команд интерпретируются как обычные адресные ЗУ соответствующей емкости.

Счетчик команд **СК** содержит адрес команды, следующей за выбранной командой. Размер команд фиксирован и равен четырем байтам, и увеличение **СК** происходит сразу на четыре на каждом такте.

Блок **РОН** состоит из тридцати двух 32□разрядных регистров с доступом через два порта чтения и два порта записи. Регистры **r0** и **r1** имеют особое назначение: **r0** всегда равен нулю, а **r1** используется для хранения адреса возврата из подпрограммы.

Фиксаторы первого и второго операндов **Ф1** и **Ф2** предназначены для выборки и временного хранения операндов исполняемой операции. Фиксатор результата **ФД** предназначен для временного хранения результата выполненной операции перед записью его в **РОН**. Если результат операции используется следующей командой программы, его значение результата операции записывается также в один или оба фиксатора операндов **Ф1** и **Ф2** для исключения простоя конвейера.

3

Коммутатор первого операнда предназначен для выбора в фиксатор **Ф1** в качестве первого операнда либо в
**СК**
содержимого
**РОН**
а или поля непосредственного операнда из команды. Выбор источника определяется значением управляющих бит “T2, T1, T0” и бита признака знакового расширения “SE”, вырабатываемых ПЛМ на основе кода команды.

Блок переходов вырабатывает признак истинности условия перехода и управляет работой коммутатора при исполнении команд передачи управления.

**АЛУ** и **сдвигатель** предназначены для выполнения арифметических и логических операций и сдвигов. Исходные операнды хранятся в фиксаторах **Ф1** и
**Ф2**
, а результат записывается в фиксатор
**ФД**
.

Счетчик тактовых циклов “СТ” подсчитывает количество тактовых циклов, затраченных на выполнение программы. Счетчик сбрасывается при рестарте программы, инкрементируется на каждом тактовом цикле процессора и останавливается при завершении программы по команде остановки.

# 3. Форматы команд процессора

Используемые в модели форматы команд в основном соответствуют форматам команд RISC□процессора Intel 80860, однако некоторые команды в учебных целях были упрощены или унифицированы. Добавлена команда **stop** для завершения выполнения программы.

Форматы команд представлены на рис. 2.

## 3.1.□□□□□ Регистровые команды

Формат команды с первым операндом в регистре:

| 11 |
|---|

| 10 |
|---|

| 0 |
|---|

| КОП |
|---|

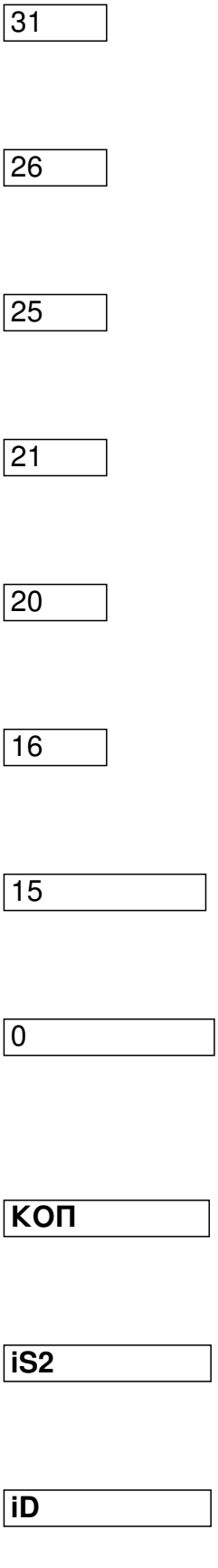
| iS2 |
|---|

| iD |
|---|

| iS | 1 |
|---|---|

| 00..0 |
|---|

Формат команды с первым операндом непосредственно в команде:



| **I1** |
|---|

Описание полей команд:

**КОП** – код операции, 6 бит;

**iS2** – номер регистра источника второго операнда, 5 бит;

**iD** – номер регистра приемника результата, 5 бит;

**iS1** – номер регистра источника первого операнда, 5 бит;

**I1** – поле непосредственного первого операнда, 16 бит.

К числу арифметических относятся команды **addu** (сложение без знака), **adds** (сложение со знаком),
**subu**
(вычитание без знака),
**subs**
(вычитание со знаком). При выполнении знаковых команд с непосредственным операндом последний расширяется до 32-х бит с учетом знака. В случае беззнаковых арифметических или логических команд – расширение непосредственного операнда беззнаковое. К логическим командам относятся
**and**
(поразрядное И),
**andnot**
(поразрядное И-НЕ),
**or**
(поразрядное ИЛИ) и

**xor**
(поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ). По результатам выполнения арифметических и логических команд устанавливается флажок условия
**C**
по следующим правилам:

- для знаковых арифметических команд **C** равен знаковому биту результата;

- для беззнаковых команд **C** соответствует биту переноса из старшего бита результата;

- для логических команд **C** устанавливается при нулевом результате.

К командам сдвига относятся **shr** (логический сдвиг вправо), **shl** (логический сдвиг влево) и **shra**
(арифметический сдвиг вправо). Используемый формат команды –
**a**
, причем поле
**iS1**
задает количество сдвигов от 0 до 31. Сдвиг на любое число разрядов выполняется в один такт.

5

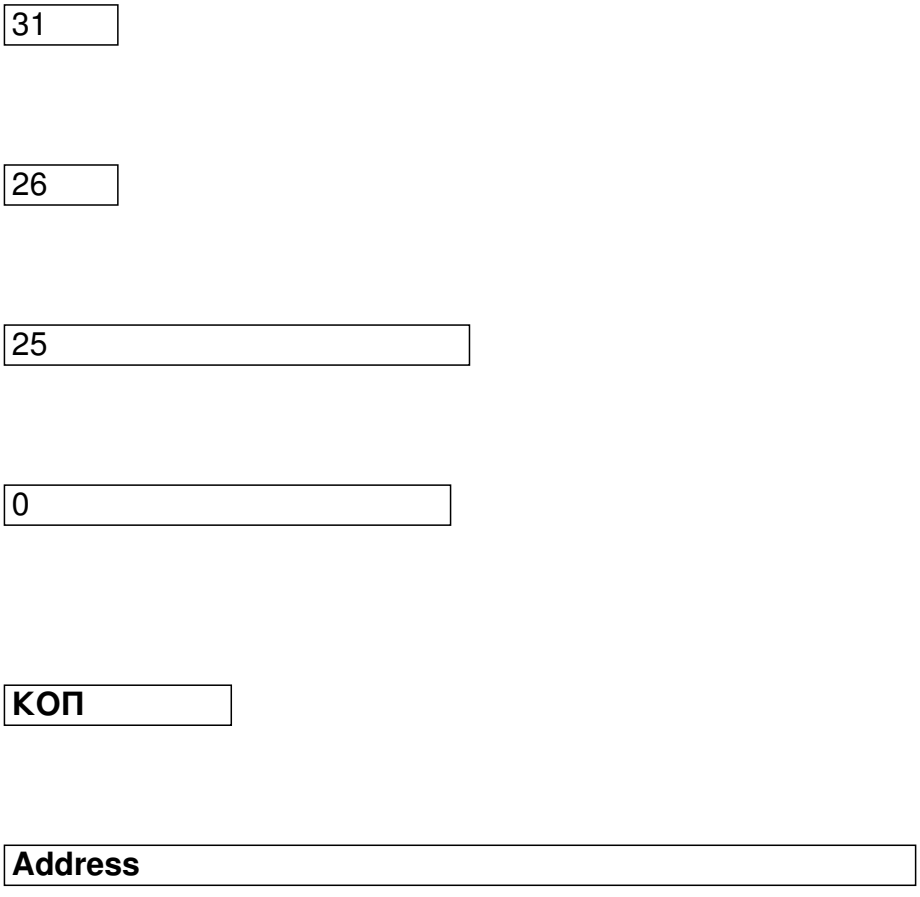
В мнемонических обозначениях регистровые команды имеют следующий вид: **com op1,op2,**
**res.**
, где com – обозначение операции, op1, op2 – первый и второй операнды, res – результат.

Первый операнд задается обозначением регистра (например: r7), десятичной константой (например: #15) или 16-ричной константой (например: 0x00FF). Для операций сдвига обозначение вида r7 означает не РОН-7, а указатель сдвига на 7 бит. Поля op2 и res всегда обозначают регистр. Напомним, что регистр r0 всегда содержит нулевое значение.

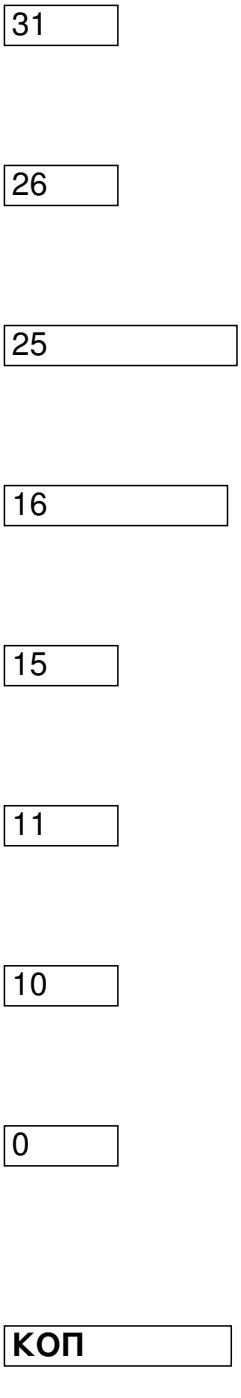
Выполнение регистровых команд происходит в 4 такта: F – выборка команд из кэша команд; D – декодирование кода операции и выборка операндов; E – исполнение; W – запись результата.

## 3.2.□□□□□ Команды перехода

Формат команды с непосредственным адресом:



Формат команды с адресом в регистре:



**00..0**

**iS1**

**00..0**

Формат команды перехода по равенству или неравенству:

31

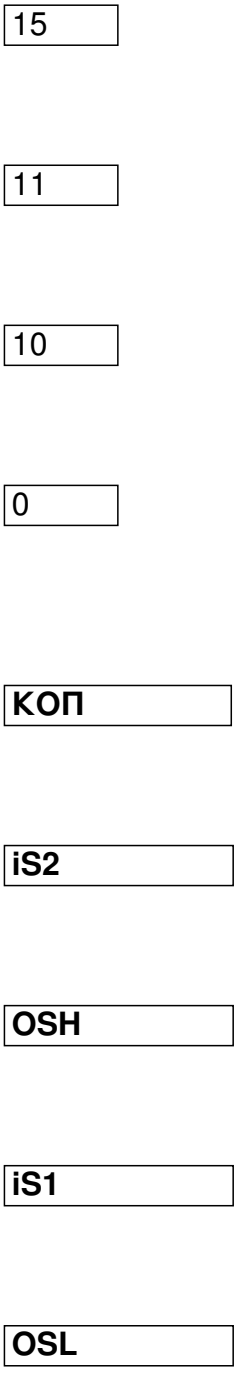
26

25

21

20

16




Описание полей команд:

**КОП** – код операции, 6 бит;

**Address** – непосредственный адрес, 26 бит;

**iS1** – номер регистра, содержащего адрес перехода либо одно из сравниваемых значений в команде перехода по равенству или неравенству, 5 бит;

**iS2** – номер регистра, содержащего второе из сравниваемых значений, 5 бит;

**OSH** – старшая часть непосредственного адреса, 5 бит;

**OSL** – младшая часть непосредственного адреса, 11 бит.

Значение адреса перехода должно быть кратно четырем и не превышать значение $2^{12}$ - 4= 0FFCh. Непосредственный адрес в команде перехода по равенству или неравенству образуется путем соединения старшей и младшей частей адреса в одно 16 разрядное поле.

Безусловные переходы **bri** и **br** имеют соответственно форматы **в** и **д** (см. рис. 2). В первом случае поле Adress содержит непосредственный адрес перехода, во втором этот адрес содержится в регистре, заданном полем iS1. Особенностью операции является то, что после команды перехода выбирается еще одна команда, а только затем – целевая команда. Такие операции называются задержанными и используются для ликвидации простоев конвейера.

Вызовы подпрограмм **cli** и **cl** выполняются аналогичным образом, но при этом в регистре r1 запоминается адрес возврата – адрес команды, второй после команды вызова. Возврат из подпрограммы реализуется командой **bri**.

Условные переходы выполняются по флажку **C** и имеют формат **в**. К ним относятся команды **bci**, **bci.t** (переход по **C** =1), **bnci** , **bnci. t** (переход по **C** =0). Суффикс **t** означает задержанную версию. Для незадержанной версии предполагается наиболее вероятным выполнение команд в естественном порядке (условие не выполняется). Для задержанной версии предполагается переход (условие выполняется). В последнем случае, как и при безусловном переходе, после команды перехода выбирается следующая по порядку, а затем целевая команда.

Если сделанное предположение не оправдывается, то возникает простой конвейера. Ошибочно выбранная команда при этом подавляется.

Особую разновидность команд передачи управления образуют команды **bei, bei.t** (переход по равенству) и **bnei, bnei. t** (переход по неравенству). Этими командами производится сравнение первого и второго регистровых операндов

и переход по результату сравнения. Команды имеют формат **r** (см. рис. 2), где iS1 и iS2 задают сравниваемые регистры, а OSH и OSL – адрес перехода.

В число команд передачи управления включена команда останова **STOP**.

## 3.3.□□□□□ Команды загрузки данных из памяти

Формат команд загрузки **ld** полностью совпадает с форматом регистровых команд (см. подраздел 4.1). При вычислении адреса производится сложение либо регистров **iS 1** и **iS 2**, либо регистра, номер которого указан в поле **iS 2**, и непосредственного операнда **I 1**. Расширение непосредственного операнда до 32 разрядов производится по тем же правилам, что и в регистровых командах. Считанный байт или слово подвергается *знаковому* расширению до двойного слова (32 разряда). Загружаемый регистр задается полем **iD**. Для загрузки возможны 4 способа адресации: косвенно-регистровая, косвенно-регистровая с индексированием, косвенно-регистровая со смещением и непосредственная. Загружаемые данные могут иметь формат байта (суффикс **. b**), двухбайтового слова (суффикс **. s**) или двойного (4-байтового) слова (суффикс

**.**
**l**
). При загрузке байтов или 2-байтовых слов они расширяются до 4-х байтов с учетом знака.

Мнемоническое обозначение команды имеет вид: **ld.f op1 (op2), res**, где суффикс f определяет размер данного (f=b, s или l); op1 – регистр индекса или непосредственный операнд (непосредственный адрес или смещение); op2 – регистр базового адреса; res – регистр-приемник.

Загрузка выполняется в 5 тактов: F – выборка команды; D – декодирование; E – исполнение (вычисление адреса и его загрузка в регистр адреса PA); R – чтение данного из кэша данных; W – запись в регистр-приемник.

## 3.4.□□□□□ Команда записи данных в память

Формат команды записи **st** совпадает с форматом команды перехода по равенству или неравенству (см. подраздел 4.2). При вычислении адреса производится сложение регистра **iS2** и расширенного непосредственного адреса

**OSH:OSL**, который рассматривается как смещение. По полученному адресу в память записывается, в зависимости от кода операции, либо младший байт, либо младшее слово, либо целиком значение регистра.

Команда записи в память **st** имеет формат **r** (см. рис. 2), где **iS1** – номер регистра-источника запоминаемого данного. Адрес равен сумме регистра, заданного

**iS2**
, и смещения
**OSH.**
**OSL**
. По вычисленному адресу в память записывается младший байт, младшее слово или полное содержимое регистра-источника. Мнемоническое обозначение команды:
**st.**
**f**
**os (**
**rs2)**
, где rs1 – регистр-источник, os – смещение, rs2 – регистр индекса.

Для операции возможны 3 способа адресации: непосредственная (rs2=r0), косвенно-регистровая (os=0), косвенно-регистровая со смещением (os<>0, rs<>r0).

Запись в память выполняется в 4 такта: F – выборка команды; D – декодирование; E – вычисление адреса; W – запись в кэш данных.

## 3.5.□□□□□ Кодирование операций и программирование ПЛМ

Двоичные коды операций приведены в табл.1. В таблице использованы следующие обозначения. Бит f для арифметико-логических команд задает формат команды: “регистр - регистр” (f=0) или “регистр - непосредственный операнд” (f=1). Для команды загрузки ld бит f=0 задает косвенно-регистровую адресацию или адресацию с индексированием (формат команды – а), бит f=1 – непосредственную адресацию или адресацию со смещением (формат команды – б). Биты dd в командах загрузки и записи в память определяют формат передаваемого данного: байт (dd=00), слово (dd=01) или двойное слово (dd=10).

Управление выработкой управляющих сигналов в модели процессора осуществляется с помощью программируемой логической матрицы (ПЛМ), которая содержит связанные матрицы И и ИЛИ. ПЛМ содержит 128 строк.

Входами ПЛМ являются 6 бит кода операции и их инверсные значения ($x_5$, $x_5$, $x_4$, $x_4$, $x_3$, $x_3$, x

2

, x
2
, x
1
, x
1
, x
0
, x
0
).

## 4.

9

# Описание работы с программой моделирования

Для запуска программы моделирования необходимо запустить исполняемый файл “I860Risc.exe”. После запуска появится окно программы, содержащее в верхней части пять кнопок: “**Процессор**”, “**Регистры**”, “**Кэш команд**”, “**Кэш данных**” и “**ПЛМ**”, предназначенных для выбора нужной вкладки модели, содержимое текущей выбранной вкладки в центре и строку состояния в нижней части окна. Строка состояния состоит из трех полей: режим исполнения команд (конвейерный или скалярный), режим трассировки программы (пошаговое или автоматическое исполнение), текущее состояние программы (бездействие, исполнение, пауза).

## 4.1. Вкладка “Процессор”

Вкладка “Процессор” содержит наглядное изображение структурной схемы процессора

и отображает процесс исполнения программы, изменение состояния регистров, фиксаторов, конвейера и флагов.

На вкладке расположены кнопки “Рестарт”, “Запустить/ Остановить/ Продолжить”, “Параметры”. При нажатии на кнопку “Рестарт” происходит перезапуск программы (аналогично сигналу Reset процессора). Кнопка “Запустить/Остановить/Продолжить” в зависимости от состояния процессора позволяет запустить программу, либо приостановить исполнение программы, либо выполнить очередной шаг программы в пошаговом режиме. Название и функция этой кнопки меняется автоматически и зависит от текущего состояния процессора и исполняемой программы. Кнопка “Параметры” выводит окно настроек параметров исполнения программы: режим исполнения (конвейерный режим, скалярный режим); режим трассировки (пошаговое исполнение, автоматическое исполнение без задержки или с задержкой от 1 до 60 секунд).

## 4.2. Вкладка “Регистры”

Вкладка “Регистры” позволяет задавать и изменять содержимое регистровой памяти процессора, а также сохранять текущее содержимое регистровой

памяти в файл на диске для последующего восстановления и восстанавливать содержимое регистров путем загрузки их из ранее сохраненного файла.

Вкладка содержит матрицу из 32 регистров процессора и кнопки “Загрузить” и “Сохранить”. Для изменения значения регистра необходимо выделить щелчком мыши или с помощью клавиш со стрелками клавиатуры нужный элемент матрицы регистров и набрать на клавиатуре новое значение регистра. Значения регистров отображаются и вводятся в виде шестнадцатеричных цифр (32-разрядное значение регистра представляется восемью шестнадцатеричными цифрами).

## 4.3. Вкладки “Кэш команд” и “Кэш данных”

Вкладка “Кэш команд” позволяет задавать и изменять содержимое памяти кэша команд, а также сохранять текущее содержимое кэша команд на диске.

Вкладка содержит матрицу из 1024 двойных слов и кнопки “Загрузить” и “Сохранить”. Для изменения значения команды по заданному адресу необходимо выделить щелчком мыши или с помощью клавиш со стрелками клавиатуры нужный элемент матрицы и набрать на клавиатуре новое значение кода команды.

Вкладка “Кэш данных” позволяет задавать и изменять содержимое памяти кэша данных, а также сохранять текущее содержимое кэша данных в файл на диске для последующего восстановления.

Вкладка содержит матрицу из 8192 байт кэша данных и кнопки “Загрузить” и “Сохранить”. Для изменения значения байта по заданному адресу необходимо выделить щелчком мыши или с помощью клавиш со стрелками клавиатуры нужный элемент матрицы и набрать на клавиатуре новое значение байта.

## 4.4. Вкладка “ПЛМ”

Вкладка “ПЛМ” позволяет задавать и изменять структуру связей программируемой логической матрицы (ПЛМ), которая осуществляет декодирова

ние команд процессора, а также сохранять текущее состояние ПЛМ в файл на диске для последующего восстановления.

Вкладка содержит макет ПЛМ (связанные матрица “И” и матрица “ИЛИ”), содержащей

128 строк, и кнопки “Загрузить” и “Сохранить”. Для установки или снятия перемычки в точке пересечения горизонтальных и вертикальных линий матриц ПЛМ необходимо произвести щелчок левой кнопкой мыши вблизи данной точки, либо выделить с помощью клавиш со стрелками нужный элемент ПЛМ и нажать клавишу “Enter” или клавишу пробела (space).

Входами матрицы “И” данной ПЛМ являются 6 бит кода операции (старшие 6 бит из 32 разрядного кода команды) и их инверсные значения.

Выходами матрицы “ИЛИ” данной ПЛМ являются 11 бит управляющих сигналов, которые можно разбить на 3 группы по их функциональному назначению.

1. Биты AS (ALU/Shift), MA (Memory access) и BR (Branch) – признаки типа команды. Они определяют тип команды и выбирают устройство для ее исполнения. Допускается установка в единичное состояние только одного из этих бит. Возможные типы команд приведены в таблице типов команд.
2. Биты T2, T1, T0 и SE – признаки, управляющие работой коммутатора и определяющие тип первого операнда и вид его расширения (знаковое или беззнаковое). Возможные типы первого операнда приведены в таблице типов первого операнда.

3. Биты DM, C2, C1, C0 – признаки, определяющие режим работы устройств и коды операций для этих устройств. Эти признаки подаются на все устройства одновременно, но воспринимает их только выбранное в данный момент устройство. Возможные коды операций приведены в таблицах кодов операций.

12

**Таблица  1. Наименование типов команд**

**Биты типа команды**

**Тип операции**

**Исполнительное устройство**

**AS**

**MA**

**BR**

1

0

0

АЛУ или сдвиг

АЛУ или сдвигатель

0

1

0

Обращение к памяти

Блок памяти и АЛУ в режиме  беззнакового сложения без изменения флагов

0

0

1

Передача управления

| | |
|---|---|
| Блок переходов | |
| Все остальные комбинации | |
| Остановка программы | |
| Нет | |

**Таблица  2. Определение типа первого операнда**

**Биты типа операнда**

**Тип первого операнда**

**T2**

**T1**

**T0**

0

0

0

Операнд из регистра с номером, заданным полем iS1

0

0

1

Непосредственный операнд из поля I1 команды

0

1

0

Непосредственное смещение из полей OSH:OSL команды

0

1

1

Непосредственный адрес из поля Address команды

1

0

0

Значение поля iS1 команды

Комбинации 101 - 111 не используются

**Таблица  3. Задание режимов работы устройств**

**Бит DM**

**Режим работы устройства**

**АЛУ и сдвигатель**

**Блок памяти**

**Блок переходов**

0

**АЛУ**

Чтение из памяти

Незадержанный переход

1

Сдвигатель

Запись в память

Задержанный переход

**Таблица  4. Кодирование арифметико-логических операций**

**Биты кода операции**

**Операция**

**Описание операции**

**C 2**

**C 1**

**C 0**

0

0

0

addu

Сложение без знака

0

0

1

adds

Сложение со знаком

0

1

0

subu

Вычитание без знака

0

1

1

subs

Вычитание со знаком

1

0

0

and

Поразрядное “И”

1

0

1

andnot

Поразрядное “И НЕ”

1

1

0

or

Поразрядное “ИЛИ”

1

1

1

xor

Поразрядное “Исключающее ИЛИ”

**Таблица  5. Кодирование операций сдвига**

**Биты кода операции**

**Операция**

**Описание операции**

**C 2**

**C 1**

**C 0**

0

0

0

shr

**Логический сдвиг вправо**

0

0

1

shl

Логический сдвиг влево

0

1

0

shra

Арифметический сдвиг вправо

Комбинации 011 - 111 не используются

**Таблица  6. Задание типа расширения первого операнда**

| **Бит SE** |
| --- |
| **Тип расширения** |

0

**Беззнаковое**

1

Знаковое

13

Таблица № 7. Кодирование операций доступа к памяти

**Биты кода операции**

**Операция**

**Описание операции**

**C 2**

**C 1**

**C 0**

0

0

0

.b

**Обращение за байтом**

0

0

1

.s

Обращение за словом

0

1

0

.d

Обращение за двойным словом

Комбинации 011 - 111 не используются

Таблица № 8. Кодирование операций перехода

**Биты кода операции**

**Операция**

**Описание операции**

**C 2**

**C 1**

**C 0**

0

0

0

br

**Безусловный переход**

0

0

1

cl

Вызов подпрограммы

0

1

0

bc

Переход при переносе

0

1

1

bnc

Переход при отсутствии переноса

1

0

0

be

Переход при равенстве

1

0

1

bne

Переход при неравенстве

Комбинации 110 - 111 не используются

**Таблица  9. Пример заполнения ПЛМ**

| КОП | Операция | Описание операции | Исп | DM | C2 | C1 | C0 | T2 | T1 | T0 | SE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 000001 | addu i1 | Сложение без знака | AS | 0 | 0 | 0 | 0 | | | | |

0

0

1

0

000010

adds reg

## Сложение со знаком
## AS

0

0

0

1

0

0

0

1

000111

subs i1

## Вычитание со знаком
## AS

0

0

1

1

0

0

1

1

001000

and reg

## Поразрядное И
## AS

0

1

0

0

0

0

0

0

a)

31

26

25

21

20

16

15

11

10

0

**КОП**

| **iS** | **2** |
|---|---|

**iD**

**iS1**

**00..0**

б)

31

26

25

21

20

16

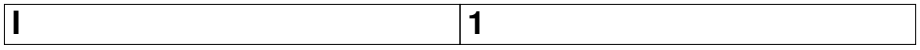
15

0

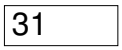

КОП

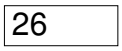

iS2


iD


I
1


в)

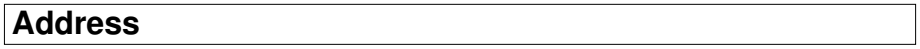

31


26

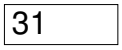

25

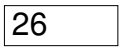

0

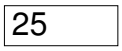

КОП

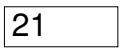

Address

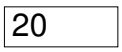

г)

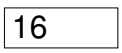

31

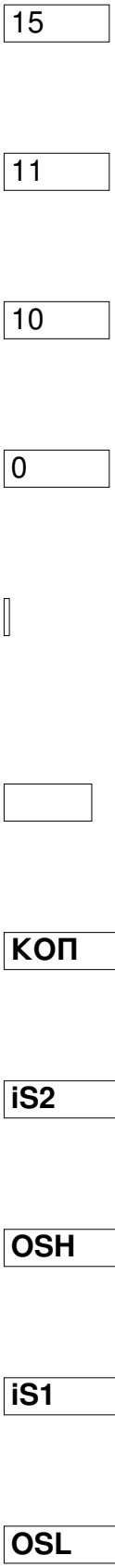

26


25


21


20


16


15
11
10
0
КОП
iS2
OSH
iS1
OSL

д)

31

26

25

16

15

11

10

0

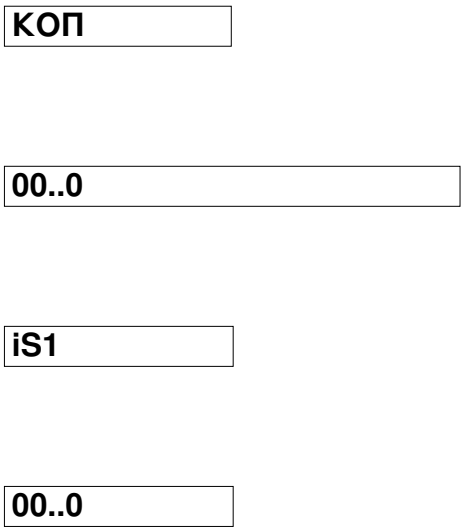



Рис. 2. Форматы команд

**5.**

**14**

# Задания по лабораторным работам

## Задания по лабораторной работе 1

### Выполнение регистровых операций и переходов

1. Вычислить наибольший делитель (НОД) двух чисел A и B по алгоритму Евклида. Указание .
A и B сравниваются между собой. Из большего числа вычитается меньшее до тех пор, пока они не сравняются между собой.

2. Вычислить остаток от деления десятичного числа на 9.Указание. Найти сумму цифр числа, а затем остаток от ее деления на 9.

3. Вычислить остаток от деления 16-ричного числа на 15=0Fh. Указание. Найти сумму 16-ричных цифр числа и остаток от ее деления на 15.

4. Найти максимальное из четырех чисел.

5. Найти минимальное из четырех чисел.

6. Умножить двоичное число B на 100.

Указание. B*100=B*(64+32+4)=B*($2^6$+$2^5$+$2^2$). Умножение на $2^n$ выполняется путем сдвига влево на n бит.

7. Умножить двоичное число B на 1000.

Указание. B*1000=B*($2^{10}$-$2^4$-$2^3$).

8. Проверить вхождение заданного байта в 4-байтовое слово. Вхождение байта кодируется единицей, не вхождение – нулем.

9. Проверка нахождения числа X в заданный диапазон (l<=X<=h, где l и h – границы диапазона).

Примечание. При программировании безусловных переходов учтите, что эти переходы задержанные, то есть после команды перехода выполняется команда, следующая по порядку, а лишь затем – целевая команда.

Для условных переходов возможно использование задержанной и незадержанной версий. Предлагается реализовать оба варианта и затем сравнить их по времени исполнения алгоритма.

15

**Задания по лабораторной работе  2**

**Операции над массивами. Подпрограммы**

Задания по этой работе заключаются в последовательной выборке элементов одного

или двух массивов и обработке этих элементов в форме подпрограммы. Результаты записываются в один из исходных массивов или в другой массив, а в некоторых случаях формируются в регистрах.

При выполнении команды загрузки ld возникает щель загрузки, которая должна быть заполнена подходящей командой, а при отсутствии таковой – командой nop. Команда вызова подпрограмм выполняется по задержанной версии.

1. Заданы два массива A[1..n], B[1..n]. Сформировать массив C[1..n], где C[i]=НОД(A[i],B[i]).

2. Сформировать массив остатков от деления элементов массива десятичных чисел на 9.

3. Сформировать массив остатков от деления элементов массива 16-ричных чисел на 15.

4. Выполнить поэлементное умножение массива на 100.

5. Выполнить поэлементное умножение массива на 1000.

6. Найти максимальное число в массиве.

7. Найти минимальное число в массиве.

8. Подсчитать число элементов массива, содержащих заданный байт.

9. Подсчитать число элементов массива, входящих в заданный диапазон.

# 6. Порядок выполнения

1. Ознакомьтесь со структурной схемой RISC-процессора i80860, а также с основными принципами его работы.
2. Разработать программу согласно номеру варианта своего задания.
3. Составить для каждой операции строку декодирования ее в ПЛМ, используя данные в таблицах 1-8. В таблице 9 представлен пример заполнения для некоторых команд.
4. Ввести программу в эмулятор и заполнить ПЛМ.
5.

16

Выполнить программу в скалярном и конвейерном режимах.

1. Составить отчет по работе, включающий следующие пункты: программу на языке ассемблера, структуру ПЛМ, программу в машинных кодах.
1. В чем отличие скалярного и конвейерного режимов работы?
2. Какие ступени конвейера содержит i80860?
3. Каковы основные черты работы RISC-процессоров?
4. Каково основное назначение ПЛМ?
5. Какие типы команд переходов существуют в i80860?

# 7. Контрольные вопросы
# Библиографический список

1. Андреев В.П., Везенов В.И., Волковыский В.Л., Пржегорлинский В.Н. Основы теории и архитектура конвейерных ЭВМ. Рязань, РРТИ, 1992.
2. Корнеев В.В., Киселев А.В. Современные микропроцессоры. Издательство “Нолидж”, 1998.
3. Архангельский А.Я. Программирование в C++ Builder 4. М, 1999.
4. 1..... Структурная схема RISC процессора i80860 1
5. 2. Описание функциональных узлов процессора..... 1

# Содержание