Лабораторная работа №7 Реализация устройства с плавающей точкой

Закрепить и расширить знания, полученные на лекциях.

1. Уяснить состав регистровой модели FPU, назначение и порядок использования регистров, организацию стека FPU.

I. Расчет учебного времени

Содержание и порядок проведения лекции	Время, мин
Вступительная часть	5
Основная часть (текст лекции)	
Введение	3
Учебные вопросы:	
1. Программная модель устройства FPU.	45
2. Организация стека устройства FPU.	30
Заключение	5
Заключительная часть	2

II. Литература

1. Е.П. Угрюмов. Цифровая схемотехника. – СПб: БХВ-Петербург, 2007.
2. Б.Я. Цилькер, С.А. Орлов. Организация ЭВМ и систем. – СПб: Питер, 2007.
3. О.М. Тарасов, Д.И. Кирик, В.И. Кабаров. Схемотехническое моделирование электронных цифровых устройств. – СПб: ВАС, 2008.

III. Учебно-материальное обеспечение

Наглядные пособия (схемы):

1. Проектор, ноутбук, слайды.

Раздаточный материал:

1. Программная модель устройства FPU.

Текст лекции

Введение

Объявить тему, цели, порядок проведения занятия.

Напомнить, что в программную или регистровую модель любого процессора включаются только те регистры, которые доступны программисту на уровне машинных команд.

Учебные вопросы

1. ПРОГРАММНАЯ МОДЕЛЬ УСТРОЙСТВА FPU

Устройство **FPU** (то есть устройство с плавающей точкой) имеет простую программную модель. Оно, как и большинство других математических сопроцессоров, опирается на общую стековую организацию. Ее применение обусловлено несколькими обстоятельствами.

Одно из них заключается в том, что в математических расчетах результат текущей операции часто может заместить один или оба исходных операнда и является операндом следующей команды. Стековая организация позволяет в этих случаях применять так называемые безадресные (*нуль-адресные)* команды небольшой длины, сокращая число обращений к памяти и, следовательно, повышая быстродействие процессора.

Представление математических выражений в обратной польской записи естественно приводит к стековой организации процессора, который вычисляет эти выражения.

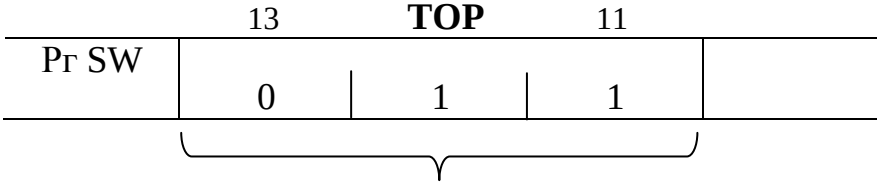
Основу программной модели устройства **FPU** образует регистровый стек из восьми 80-битных регистров данных **R0 — R7**, называемых также численными или арифметическими регистрами.

Стек представлен на рисунке:

Программная модель устройства **FPU**

В восьми 80-битных регистрах данных **R0 — R7** хранятся числа, представленные в форме с плавающей точкой в расширенном вещественном формате *(ранее этот формат назывался временным вещественным форматом)*. Такой формат является единственным внутренним форматом чисел в устройстве **FPU**.

В любой момент времени трехбитное поле **TOP** (***TOP of stack***) в слове состояния **SW** определяет регистр, являющийся текущей вершиной стека и обозначаемый ***ST(0)*** или просто **ST** (***Stack Top***).



Если, например, в поле **TOP** содержится **011B**, то вершиной стека будет регистр **R3** и именно он указывается в командах как ***ST(0)*** или **ST**. Регистр **R4** находится сразу «ниже» регистра **ST** и обозначается ***ST(1)***. Регистр **R2** оказывается в самом «низу» стека и обозначается ***ST(7)***, так как регистры адресуются по кругу, независимо от их порядковых номеров, начиная с вершины стека.

Другими словами, регистр **ST(0)** содержит последнее включенное в стек значение, регистр **ST(1)** - предпоследнее и т. д.

Операция загрузки или включения в стек *(push*) осуществляет декремент поля **TOP** и загружает адресуемые данные в новую вершину стека, что приводит к автоматической перенумерации регистров. В операции сохранения и извлечения из стека *(pop)* в получатель, например в память, передается содержимое текущей вершины стека, а затем производится инкремент поля **TOP**.

Таким образом, в стандартных стековых операциях поле **TOP** выполняет функции традиционного указателя стека **SP** *(Stack Pointer)*.

Тэги характеризуют содержимое каждого из регистров стека.

Остальными регистрами в программной модели устройства **FPU** являются **16**-битные ***регистры управления и состояния***, а также два **48**-битных регистра ***указателей команды и операнда***. Информация, содержащаяся в регистрах-указателях команды и операнда (***совместно их иногда называют указателями особого случая***), предназначена только для процедур обработки особых случаев.

Когда устройство **FPU** выполняет численную команду, оно автоматически сохраняет в этих регистрах адреса численной команды и ее операнда (*если он есть*), а также первые **11** бит кода операции.

Регистры-указатели доступны только в командах сохранения среды **FSTENV**, сохранения полного состояния **FSAVE**, загрузки среды **FLDENV** и восстановления, т. е. загрузки, полного состояния **FRSTOR**.

Средой устройства FPU называется содержимое всех регистров (управления, состояния, тэгов и обоих указателей), **кроме стека.**

Полное состояние – содержимое всех регистров (и среды, и стека).

Команды:

FSTENV – запись среды в область памяти с начальным адресом, указанным в команде.

FSAVE – запись полного состояния в область памяти с начальным адресом, указанным в команде.

FLDENV – возвращение среды в регистры **FPU**.

FRSTOR – возвращение полного состояния в регистры **FPU**.

При копировании среды в память информация, в зависимости от режима (R или P), а также в зависимости от разрядности операндов (32- или 16-битные) хранится в одном из четырех форматов:

- 32-битный формат P-режима;
- 32-битный формат R -режима;
- 16-битный формат P-режима;
- 16-битный формат R -режима;

В **P**-режиме указатели команды и данных не содержат кода операции, поэтому при необходимости обработчик особого случая считывает код операции из памяти.

На рисунке представлен пример хранения в памяти среды в 32-битном формате Р-режима:

31 23	15 7 0	
*	Селектор операнда	18H
Смещение операнда		14H
*	Селектор CS	10H
Смещение IP		CH
*	Слово тэгов	8H
*	Слово состояния	4H
*	Слово управления	0

* Зарезервированы

Рис.1.

В ячейках с адресами CH и 10H хранится указатель команды, в ячейках с адресами 14H и 18H – указатель данных (см. программную модель FPU).

Команда сохранения полного состояния **FSAVE** в дополнение к среде сохраняет в памяти содержимое всех регистров стека в **16**- или **32**-битном формате.

Таким образом, для сохранения полного состояния в **32**-битном формате требуется область памяти из ***108 байт*** (80 байт в стеке + 28 байт на рис. 1), а в **16**-битном – из ***94 байт*** (80 байт в стеке + 14 байт (по 2 байта) на рис. 1).

Благодаря информации, сохраняемой в памяти командами **FSTENV** или **FSAVE**, процедура обработки особого случая может точно определить команду и операнд, вызвавшие особый случай, установить причину его возникновения и предпринять соответствующие корректирующие действия.

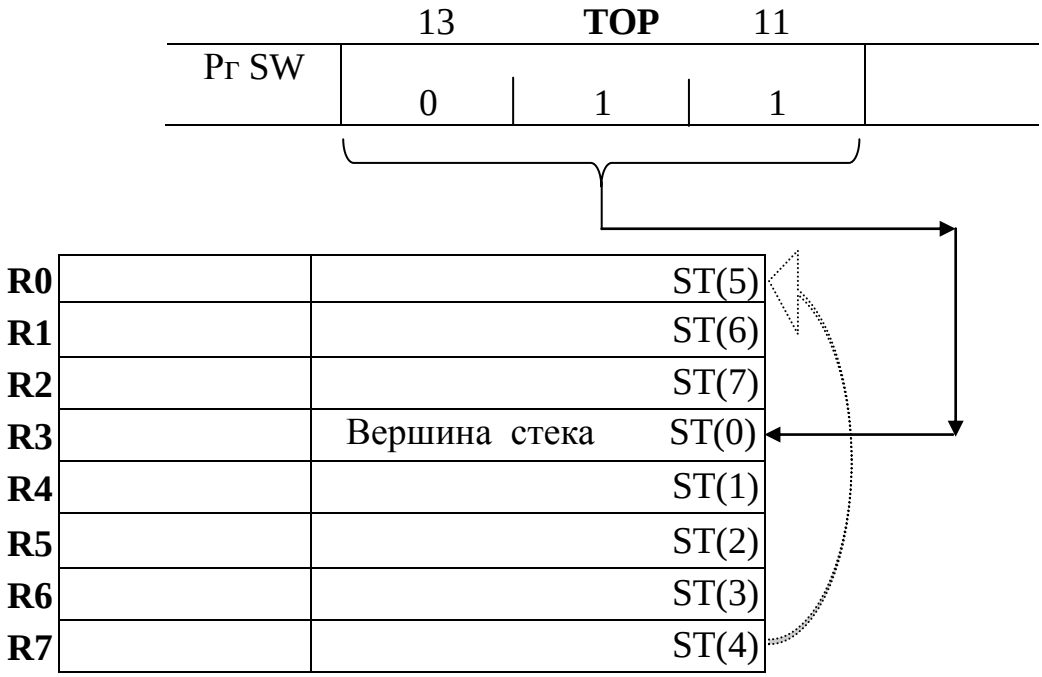
2. ОРГАНИЗАЦИЯ СТЕКА УСТРОЙСТВА FPU

Многие численные команды имеют несколько режимов адресации, позволяющих программисту неявно оперировать вершиной стека или явно определять конкретные регистры относительно **TOP**.

Ассемблер **ASM 386/486** поддерживает эти режимы адресации регистров, используя обозначения **ST(0)** или просто **ST** для текущей вершины стека и **ST(i)** для **i**-го регистра в стеке ($0 \le i \le 7$).

Если, например, **TOP** содержит **011B** (вершиной стека является регистр 3), то команда:

FADD ST,ST(2) суммирует содержимое двух регистров стека FPU (регистров 3 и 5):



Отличия регистрового стека FPU от классического стека, реализуемого в памяти

В организации регистрового стека устройства **FPU** имеется несколько отличий от классического стека, который в большинстве современных процессоров аппаратно реализуется в памяти.

Во-первых, стек имеет круговую (кольцевую) организацию.

Это означает, что если поле **TOP** содержит **000B** и производится его декремент, новым содержимым **TOP** будет **111B**, а если поле TOP содержит **111B** и осуществляется его инкремент, новым содержимым будет **000B**.

Контроль за использованием стека должен осуществлять программист. В частности, необходимо помнить, что максимальное число включений в стек без промежуточных извлечений равно **8,** а девятое включение «перезапишет» значение, включенное в стек первым. (На самом деле сопроцессор зафиксирует здесь особые случаи нарушения стека и недействительной операции и «перезаписывание» произойдет, если только прерывание от этих особых случаев замаскировано или запрещено.)

Во-вторых, в численных командах допускается явное или неявное обращение к регистрам стека с модификацией или без модификации поля TOP.

Так, в унарных операциях операндом служит содержимое вершины стека, а результат замещает операнд. Например, команда **FSQRT** замещает число из вершины стека значением корня квадратного из этого числа.

В некоторых бинарных операциях операндами служат числа в двух верхних регистрах стека, а результат помещается на место одного из них. Более того, команда сравнения **FCOMPP** сравнивает содержимое двух верхних регистров стека и удаляет его, производя инкремент **TOP** на **2.** Наконец, в бинарных операциях допускается явное указание регистров, содержащих операнды.

Программист работает относительно текущей вершины стека и не употреб-

ляет абсолютных имен регистров **RO, R1** и т. д. Принятые в устройстве **FPU** организация стека и адресация регистров упрощают программирование с использованием подпрограмм. Через стек удобно передавать подпрограмме параметры и осуществлять доступ к ним в процессе ее выполнения, а также обращаться к результатам после завершения подпрограммы.

Применяя для передачи параметров стек, а не специализированные регистры, вызывающие программы получают большую гибкость в использовании стека. До тех пор, пока стек не заполнен, каждая программа перед вызовом численной подпрограммы просто загружает параметры в стек.

После этого подпрограмма адресует свои параметры как **ST, ST(1)** и т. д., несмотря на то, что в одном вызове **TOP** относится, например, к физическому регистру **R3**, а в другом — к физическому регистру **R5.**

В третьих, устройство FPU имеет команды, в которых не выдерживаются обычные соглашения о стеке.

Например, команда **FST** *(сохранение в памяти)* передает содержимое вершины стека в память, но не производит инкремента поля **TOP**. Другая команда **FXCH** *(обмен)* позволяет обменять содержимое вершины стека и любого другого регистра.

С каждым регистром стека ассоциируется двухбитный **тэг** *(признак)*, совокупность которых образует ***слово тэгов*** **TW** ***(Tag Word*****).**

Тэг регистра **R0** находится в младших битах этого слова, а тэг регистра **R7** в старших битах (см. *рис. 2*).

Тэг в общем виде показывает, что содержится в соответствующем регистре и помогает устройству **FPU** в выполнении команд. Содержимое тэга интерпретируется следующим образом:

00 - допустимое ненулевое число,
01 - нуль,
10 - специальное значение (нечисло, не поддерживаемый формат, бесконечность или денормализованное число),
11 - пустой регистр.

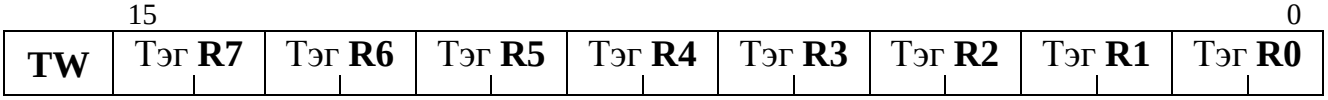


Рис. 2. Формат слова тэгов TW

Тэг с кодом **11B** показывает отсутствие данных; в этом случае регистр называется пустым и одно из действий устройства **FPU** при инициализации заключается в загрузке в биты тэгов всех регистров кода **11B.**

Попытка команды извлечь число из пустого регистра фиксируется как особые случаи нарушения стека и недействительной операции. Кроме того, попытка загрузить число в непустой регистр также вызывает регистрацию аналогичных особых случаев.

Программист может использовать ***слово тэгов*** для упрощения интерпретации содержимого регистров, без сложного декодирования фактических данных

(разумеется, после передачи этого слова в память).

Так как размещение тэгов соответствует физическим регистрам **R0** — **R7**, для ассоциирования тэгов с относительными регистрами **ST(0)** — **ST(7)** потребуется привлекать ***поле TOP*** из слова состояния.

Точные значения тэгов образуются при выполнении команд сохранения среды **FSTENV** и сохранения полного состояния **FSAVE** в соответствии с фактическим содержимым непустых регистров стека. При выполнении других команд устройство **FPU** модифицирует слово тэгов, только показывая пустой или непустой регистр стека.