Окончание. Начало в № 9'2003, 1'2004.

Стековые процессоры, или Новое – это хорошо забытое новое

Иосиф Каршенбойм

iosifk@narod.ru

TF16 фирмы «Технофорт»

После исторических исследований, заморских диковин и патриархов Форт-индустрии настала пора показать то, что сделано у нас в России, в Петербурге, на Васильевском острове. Материалы, представленные в данном разделе, предоставлены фирмой «Технофорт», фрагменты программ предоставлены ведущим разработчиком процессора TF16 А. Капрановым, сотрудником фирмы «Технофорт».

Процессор TF16 фирмы «Технофорт» выполнен как стековый процессор и предназначен для встроенных применений. Вместе с периферией, размещаемой на кристалле FPGA, данный процессор представляет собой полную систему на кристалле, блок-схема которой приведена на рис. 15. Примерами применения Форт-систем с процессором TF16 могут служить интеллектуальные приборы «Скиф» (рис. 12). Данные приборы представляют собой малогабаритный переносной микрокомпьютер со встроенной операционной системой реального времени и с поддержкой файловой системы, совместимой с MS-DOS. Приборы имеют память до 2 Мбайт. Питание прибора осуществляется от аккумуляторов. «Скиф» используются для решения многочисленных задач автоматизации в условиях автономной работы как терминал сбора данных, универсальный регистратор и управляющая машина, заменяя там, где это возможно, дорогие и крупногабаритные контроллеры на базе промышленных PC.

Но наибольшего внимания заслуживает применение Форт-системы с процессором TF16 в качестве кассового регистратора, как наиболее массового изделия. Применение системы на кристалле становится экономически целесообразным, учитывая то, что фирма Altera разработала технологию HardCopy, когда любой проект из существующего проекта для FPGA может быть переведен в ASIC. Сегодняшний кассовый регистратор представляет собой многофункциональный сетевой терминал, способный выполнять как функции кассового регистратора, так и другие функции, например терминала обработки данных на складе, сетевого терминала справочной службы, устройства для маркировки товара и т. д.

Рис. 12. Интеллектуальный прибор «Скиф»

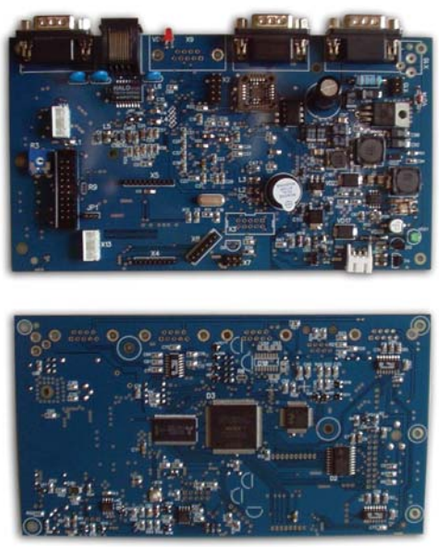

Рис. 13. Плата кассового регистратора

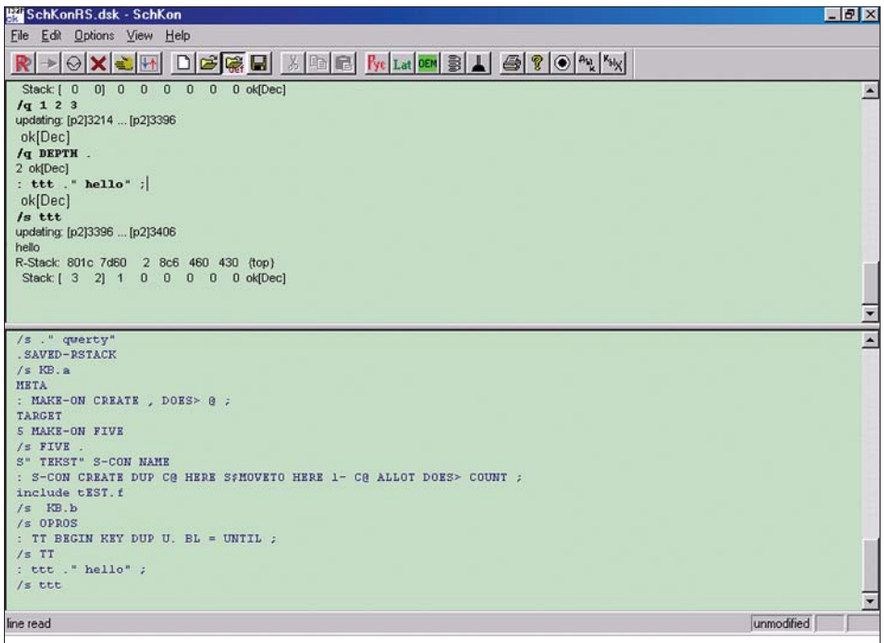

Рис. 14. Основное окно программы ТФ32

На рис. 13 приведена фотография платы кассового регистратора. Кассовый регистратор имеет встроенную операционную систему реального времени, имеет поддержку файловой операционной системы и поддержку сетевых протоколов TCP/IP. Кассовый регистратор имеет несколько разъемов для подключения сканеров штрих-кода и разъем для подключения к сети Ethernet 10/100. Информация выводится на цифробуквенный ЖКИ — дисплей.

Для разработки и отладки ПО, фирмой «Технофорт» разработана программа ТФ32, позволяющая производить отладку полностью собранного кассового регистратора. Регистратор по последовательному каналу связи подключается к хост-компьютеру.

Запускается программа ТФ32. Основное окно программы показано на рис. 14.

Монитор пользователя

Монитор пользователя — это программа, при помощи которой TF16 поддерживает режим обмена с хост-компьютером. Данная программа так же разработана в фирме «Технофорт». Монитор — это программа, зашиваемая в память программ, расположенную на кристалле. При включении питания процессор TF16 начинает выполнять программу монитора, предназначенную для загрузки, выполнения и записи на Flash-диск пользовательских программ. Если Flash-диск содержит программу, монитор загружает ее (первые 64 Кбайт диска копируются в сегмент ОЗУ 28h и управление передается на точку входа). В противном случае, если Flash-диск отсутствует (или пустой), монитор переходит на программу обмена с компьютером.

Команды протокола (содержатся в целевом Форт-компиляторе):

- TEST_INIT — инициализация работы.
- TEST_LOAD — загрузка программы в сегмент памяти 28h.
- TEST_RUN — выполнить программу с заданного адреса.
- TEST_SAVE — записать программу на Flash-диск.
- RSRUN (adr --) — загрузить программу в ОЗУ и выполнить. Adr — стартовый адрес программы.
- RSSAVE (adr --) — загрузить программу в Flash-диск и выполнить. Adr — стартовый адрес программы.

Итак, переходим непосредственно к процессору.

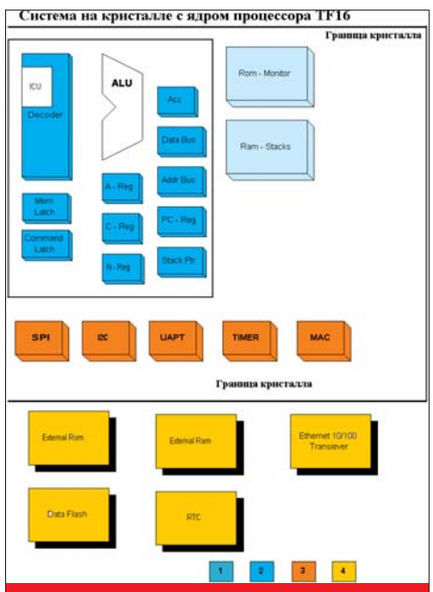

Рис. 15. Блок-схема системы на кристалле с ядром процессора TF16.
Цветом «1» показаны ресурсы ядра процессора,
цветом «2» — блоки памяти, расположенные на кристалле,
цветом «3» — периферийные устройства, расположенные на кристалле,
цветом «4» — внешние периферийные устройства и блоки памяти, находящиеся вне кристалла.

Ресурсы процессора TF16

Регистры процессора представлены в таблице 6.

Таблица 6. Регистры процессора

Название	Описание	При установке сигнала «Сброс» регистры принимают следующие значения
a1, a2	16-разрядные регистры общего назначения. Регистры a1-a2 могут использоваться в качестве базовых при обращении к памяти. Используются для вычислений.	
t	16-разрядный аккумулятор. Для языка Форт — верхний элемент стека параметров	
n	16-разрядный регистр — расширение аккумулятора. Для языка Форт — второй элемент стека параметров.	
c	16-разрядный регистр общего назначения (счетчик цикла)	
rsp	16-разрядный указатель стека возвратов	
psp	16-разрядный указатель стека параметров	
cs, ss, ds, es	6-разрядные сегментные регистры. Позволяют работать с большим объемом памяти. кодовый сегмент стековый сегмент сегмент данных сегмент данных	0
cntr	16-разрядный управляющий регистр	0
intr	7-разрядный регистр сегмента прерываний	0
pc	16-разрядный счетчик команд	0
l	16-разрядная защелка команд	

Карта **памяти** процессора представлена в таблице 7. Полный 20-разрядный адрес A[19..0] формируется следующим образом:
A[19..16] — значение cs, ss, ds, es
A[15..0] — значение pc, l, psp, rsp, a1-a2, t

Таблица 7. Карта памяти процессора

Полный адрес	Назначение памяти	Расположение
000000-0007FF	2 Кб — ПЗУ (монитор)	Внутрикристальное ПЗУ
000800-000BFF	1 Кб — ОЗУ (стеки)	Внутрикристальное ОЗУ
280000-2FFFFF	256 K×16 — ОЗУ (может быть расширена до 512 K×16)	Внешнее ОЗУ
Flash-диск адресуется через порт как диск	4 Мб — Flash-диск (может быть добавлен)	Внешнее ПЗУ

Расположение **стеков** в адресном пространстве определяется указателями PSP, RSP и сегментным регистром SS. Указатель содержит адрес верхнего элемента стека. Направление стека определяется младшим битом регистра-указателя (0 — стек растет вверх, 1 — стек растет вниз).

Ядро процессора TF16 содержит 8-канальный контроллер **прерываний**. Типовое использование для применения в составе кассового регистратора следующее — задействованы три прерывания — от UART1, UART2 и от таймера. Их вектора прерываний соответственно — 100h, 140h, 180h. Механизм обработки прерывания (на примере таймера) приведен ниже.

```
IF cntr15 THEN                  (общая маска прерываний)
  IF ov & ie THEN               (разрешено прерывание от таймера
                                 и бит переполнения установлен)
    cntr15 = 0                  (запрет прерывания)
    cs[5..0] <--> intr[6..1]    (сохраняем текущий cs и устанавливаем
                                 в cs значение intr)
    intr0 = carry               (сохраняем значение carry)
    call 0180h                  (вызов обработчика прерываний)
  ENDIF
ENDIF
```

Возврат из прерывания происходит по команде ireturn:

```
intr[6..1] <--> cs[5..0]     (восстанавливаем значения intr и cs)
carry = intr0                (восстанавливаем значение carry)
cntr15 = 1                   (разрешаем прерывания)
return                       (возвращаемся в программу)
```

Сравнение **ресурсов, занимаемых ядрами встраиваемых процессоров на кристалле**, приведено в таблице 8.

Таблица 8. Сравнение ресурсов, занимаемых ядрами встроенных процессоров на кристалле

Название ядра	Число ячеек	Блоки встроенной памяти	Тактовая частота МГц	Микро-схема
Nios 3.0 — 16 бит	1500		Более 125	Stratix
Nios 2.0 — 16 бит	1100	7	33	EP20K200E
TF16 — 16 бит	**900**	**6**	**33**	EP1K30

Из сравнения данных, приведенных в таблице 9, видно, что ядро процессора TF16 занимает значительно меньше ресурсов, чем сопоставимые 16-разрядные ядра.

К микропроцессору могут быть подключены следующие расположенные на кристалле **периферийные устройства**:

- Ethernet MAC;
- UART — 2;
- 16-разрядный таймер;
- SPI для связи с Flash-диском;
- Порт для подключения ЖКИ;
- 2 универсальных 16-разрядных порта;
- 9-разрядный универсальный порт;
- порт управления звукоизлучателем;
- порт управление часами.

Канал связи с периферией представляет собой шину адреса, две шины данных (одна для выходящих из процессора данных, другая — для данных, поступающих из периферии в процессор) и сигналов управления. Таким образом, к данному ядру может быть подключена любая периферия из готовых блоков, находящихся в распоряжении пользователя. В зависимости от конкретных требований проекта периферийные блоки могут быть выполнены полностью аппаратно или они могут быть частично программными. Так например, блок для связи с UART может быть сделан частично программно, то есть процессор формирует диаграмму передачи побитно и передает (принимает) данные через параллельный порт.

Система команд процессора TF16

Система команд включает 4 класса:

- команды вызова подпрограмм и команды возврата — CALL/RETURN;
- команды переходов (8 команд);
- команды записи-чтения памяти (непосредственно, через индексные регистры, с инкрементом или декрементом адреса);
- математические команды — эти команды могут сопровождаться активизацией одного из двух стеков (команды языка Форт);
- команды обращения к портам.

Внутренняя система сегментации позволяет адресовать 4 Мбайт внешней памяти.

Наличие резервных кодов позволяет существенно расширить систему команд и увеличить вычислительную производительность процессора (однотактное умножение, команды поддержки арифметики с плавающей точкой и т. д.). Внутреннее регистровое пространство может быть увеличено до 256 регистров. Изменение разрядности сегментных регистров (до 16 разрядов) дает возможность получать необходимое адресное пространство (до 4 Гбайт).

Таблица 9. Базовые команды

Мнемоника	Параметры	Ключи	carry	Описание
2nop				nop nop
add	a1-2,n,c	push pop ret reg!	+	Сложение t с регистром
add	Literal		+	Сложение t с литералом
addc	a1-2,n,c	push pop ret reg!	+	Сложение с переносом t и регистра
addc	Literal		+	Сложение с переносом t и литерала
and	a1-2,n,c	push pop ret reg!		Логическая «И» t и регистра
and	Literal			Логическая «И» t и литерала
bc	label{-128:127}			Переход по смещению, если carry=1
bcp	label{-128:127}			Переход по смещению, если carry=1. Pop PS
bgez	label{-128:127}			Переход по смещению, если t>=0
bgezp	label{-128:127}			Переход по смещению, если t>=0. Pop PS
blz	label{-128:127}			Переход по смещению, если t<0
blzp	label{-128:127}			Переход по смещению, если t<0. Pop PS
bnc	label{-128:127}			Переход по смещению, если carry=0
bncp	label{-128:127}			Переход по смещению, если carry=0. Pop PS
bnz	label{-128:127}			Переход по смещению, если t<>0
bnzp	label{-128:127}			Переход по смещению, если t<>0. Pop PS
br	label{-128:127}			Безусловный переход по смещению
bz	label{-128:127}			Переход по смещению, если t=0
bzp	label{-128:127}			Переход по смещению, если t=0. Pop PS
call	Addr16			Вызов подпрограммы по 16-разрядному адресу
cycl	label{-128:127}			Переход по смещению, если c<>0, c=c-1
dsla		push pop ret	+	Двойной арифметический сдвиг влево пары t и n
dslc		push pop ret	+	Двойной циклический сдвиг влево пары t и n
dsll		push pop ret	+	Двойной логический сдвиг влево пары t и n
dsra		push pop ret	+	Двойной арифметический сдвиг вправо пары t и n
dsrc		push pop ret	+	Двойной циклический сдвиг вправо пары t и n
dsrl		push pop ret	+	Двойной логический сдвиг вправо пары t и n
dstep	a1-2,c	push pop ret reg!	+	Шаг беззнакового деления (t,n) и регистра
dstep	Literal		+	Шаг беззнакового деления (t,n) и литерала
ireturn				Возврат из прерывания
jump	Addr16			Переход по абсолютному 16-разрядному адресу
load	a1-2,n,c,cs,ss, ds,es,psp,rsp,cntr,intr	push pop ret popnt		Загрузка регистра значением t
load	Memory	-t -n		Загрузка в память значения t или n
mova	a1-2,n,c,cs,ss, ds,es,psp,rsp,cntr,intr	push pop ret reg!		Загрузка в t значения регистра
mova	Literal			Загрузка в t значения литерала
mova	Memory	-t -n		Загрузка в t или n значения из памяти
mstep	a1-2,c	push pop ret reg!	+	Шаг умножения n и регистра
mstep	Literal		+	Шаг умножения n и литерала
nop		push pop ret popnt		
or	a1-2,n,c	push pop ret reg!		Логическая «ИЛИ» t и регистра
or	Literal			Логическая «ИЛИ» t и литерала
ppop				Pop стека параметров [ps]->n->t
ppush				Push стека параметров t->n->[ps]
rdec				Декремент указателя стека возвратов

Таблица 9. Базовые команды (окончание)

Мнемоника	Параметры	Ключи	carry	Описание
return				Возврат из подпрограммы
rinc				Инкремент указателя стека возвратов
rload				Загрузка t по адресу указателя стека возвратов
rmova				Загрузка в t из адреса указателя стека возвратов
rpop				rmova + pop стека возвратов
rpush				rload + push стека возвратов
sbb	a1-2,n,c	push pop ret reg!	+	Вычитание t из регистра
sbbb	Literal		+	Вычитание t из литерала с займом
sla		push pop ret	+	Арифметический сдвиг влево t
slc		push pop ret	+	Циклический сдвиг влево t
sll		push pop ret	+	Логический сдвиг влево t
sra		push pop ret	+	Арифметический сдвиг влево t
src		push pop ret	+	Циклический сдвиг влево t
srl		push pop ret	+	Логический сдвиг влево t
sub	a1-2,n,c	push pop ret reg!	+	Вычитание регистра из t
subb	Literal		+	Вычитание литерала из t с займом
swab		push pop ret		Перестановка байтов t
xchg	a1-2,n,c,cs,ss, ds,es,psp,rsp,cntr,intr	push pop ret		Обмен содержимым t и регистра
xor	a1-2,n,c	push pop ret reg!		Логическая «Исключающее ИЛИ» t и регистра
xor	Literal			Логическая «Исключающее ИЛИ» t и литерала

Таблица 10. Команды для работы с памятью

Мнемоника	Параметр1	Параметр2	Ключи	Описание
load	(a1), (a2)	[cs], [ss], [ds], [es]	–t –n	Загрузка в память по адресу ([xs],ax) t или n
load	–(a1),–(a2)	[cs], [ss], [ds], [es]	–t –n	Загрузка в память по адресу ([xs],ax–2) t или n ax=ax–2
load	(a1)–,(a2)–	[cs], [ss], [ds], [es]	–t –n	Загрузка в память по адресу ([xs],ax) t или n ax=ax–2
load	+(a1),+(a2)	[cs], [ss], [ds], [es]	–t –n	Загрузка в память по адресу ([xs],ax+2) t или n ax=ax+2
load	(a1)+,(a2)+	[cs], [ss], [ds], [es]	–t –n	Загрузка в память по адресу ([xs],ax) t или n ax=ax+2
load	mem	[cs], [ss], [ds], [es]	–t –n	Загрузка в память по адресу ([xs],t) t или n
mova	(a1), (a2)	[cs], [ss], [ds], [es]	–t –n	Загрузка в t или n из памяти по адресу ([xs],ax)
mova	–(a1),–(a2)	[cs], [ss], [ds], [es]	–t –n	Загрузка в t или n из памяти по адресу ([xs],ax–2) ax=ax–2
mova	(a1)–,(a2)–	[cs], [ss], [ds], [es]	–t –n	Загрузка в t или n из памяти по адресу ([xs],ax) ax=ax–2
mova	+(a1),+(a2)	[cs], [ss], [ds], [es]	–t –n	Загрузка в t или n из памяти по адресу ([xs],ax+2) ax=ax+2
mova	(a1)+,(a2)+	[cs], [ss], [ds], [es]	–t –n	Загрузка в t или n из памяти по адресу ([xs],ax) ax=ax+2
mova	mem	[cs], [ss], [ds], [es]	–t –n	Загрузка в t или n из памяти по адресу ([xs],t)

Таблица 11. Ключи-модификаторы

Ключ	Описание
ret	Команда сопровождается возвратом из подпрограммы
regl	Команда с операндом-регистром сопровождается записью в этот регистр значения аккумулятора
push	Команда сопровождается push стека параметров (n->[psp])
pop	Команда сопровождается pop стека параметров ([psp]->n)
popnt	Команда сопровождается pop стека параметров ([psp]->n->t)
-t	Загрузка в память значения t
-n	Загрузка в память значения n

Команды процессора TF16 (табл. 9–11), как и у всех стековых процессоров, очень плотно упакованы и за одну команду позволяют выполнить несколько непротиворечивых действий одновременно. Далее на примерах показано, как и какие действия выполняют команды микропроцессора TF16.

Кодировка системы команд

В скобках приведено количество тактов, требуемых для выполнения данной команды.

Команды вызова: Call (3)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit address															0

Память: запись-чтение (2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	RSP		mode		a-reg		P	Sreg			D	r/w	0	1	1

Mem	Reg	A1	N	Cs	T	r/w=1 write
@!	Reg+	A2	Y	Ss	N	
@!+–	Reg–	A3		Ds		
+/–	Mem	A4		Es		

Здесь:

D — источник или приемник данных (регистр T или N).

Sreg — выбор сегментного регистра.

P — выбор (a1)+ или +(a1).

Mode: Reg: (a1):

- Reg+: +(a1), (a1)+;
- Reg-: -(a1), (a1)-;
- Mem: RSP active.

RSP: если Mode=11 (Mem):

- Mem: mem;
- @!: чтение (запись) из стека возвратов;
- @!+-: push (pop) стека возвратов;
- +/-: inc (dec) указателя стека возвратов.

RSP: если Mode<>11, 4 старших разряда задают смещение от базового регистра.

Примеры:

```
0000 1100 0000 0011   0C03    mova mem [cs] -t
0010 1100 0011 0011   2C13    rpop [ss] -n
0010 0001 0100 1011   214B    load 2 (a2) [ds] -t
0000 0111 1111 1011   07EB    load +(a4) [ss] -n
```

Работа с 16-разрядным литералом (2/3)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15 0
0	0	0	0	Mode		Sreg		Alu 2op				1	1	0	1	Const16

0	0	1 0 команда АЛУ с длинным литералом
0	0	1 1 команда АЛУ с длинным литералом + push SP
0	1	sreg команда АЛУ с содержимым по непоср. адресу
1	0	sreg запись в память по непоср. адресу

Примеры:

```
022D 1000     add 1000h      t-reg+1000 -> t-reg
```

Команда перехода по адресу (2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15 0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	Addr16

Примеры:

```
000D 1000     jump 1000h     1000 -> PC
```

Переходы (1/2/3)

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

Биты 7, 6, 5 и 4 могут принимать значения, указанные в таблице 12.

Примеры:

```
FF05    1$:   br 1$ (зацикливание — команда stop)
01A5          bncp 2$ (pop SP → n-reg → t-reg)
              nop
        2$:   nop
```

Таблица 12

0000	uncond	0	0100	T>=0	4	1000	--	8	a100	T>=0	C
0001	cycle I<>0	1	0101	T<0	5	1001	--	9	a101	T<0	D
0010	not carry	2	0110	T=0	6	a010	not carry	A	a110	T=0	E
0011	carry	3	0111	T<>0	7	a011	carry	B	a111	T<>0	F

Displacement — число со знаком {–128:127} A — активизация стека параметров (pop)

Таблица 13

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
stack		Mode		Register				alu 2op/1op				0	0	0	1

Stack:	mode:	Registers:						ALU 2-op				ALU 1-op (Shifter)	
no active	reg! T->reg	0000 cs	0100 a1	1000 psp	1100 n	0000 Dstep 0	1000 REG	0000 2/L	1000 2*AL
RET	R! N->T->r	0001 ss	0101 a2	1001 rsp	1101 c	0001 Mstep 1	1001 AND	0001 2/A	1001 2*C
pop->N	reg@	0010 ds	0110 a3	1010 intr	1110	0010 + 2	1010 OR	0010 2/C	1010 --
T->N->push	alu1op	0011 es	0111 a4	1011 cntr	1111	0011 +c 3	1011 XOR	0011 --	1011 --
						0100 s– 4	1100 SWAB	0100 D2/L	1100 D2*AL
						0101 s–c 5	1101 --	0101 D2/A	1101 D2*C
						0110 – 6	1110 --	0110 D2/C	1110 -- ,
						0111 –c 7	1111 NOP	0111 --	1111 NOP

ALU: +c, s-c, -c — команды с переносом; s- — обратное вычитание (reg — t); - — прямое вычитание (t — reg);
Сдвиги: L — логический, A — арифметический, C — циклический (без переноса).

8-разрядный литерал (1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte literal								Alu 2op				1	0	0	1

Byte literal — беззнаковое число

Примеры:

```
0129    add 1    (inc)
0169    sub 1    (dec)
```

Математика, сдвиги, стек параметров, return (1/2/3) (табл. 13)

Примеры:

```
0000  0100  1100  0001    04C1    load a1
0100  0100  1100  0001    44C1    load a1 ret
1000  0100  1100  0001    84C1    load a1 pop
1100  0100  1100  0001    C4C1    load a1 push
0011  XXXX  0000  0001    3X01    srl
0010  1101  0000  0001    2D01    dstep c
```

Команды обращения к портам (1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS												r/w	1	1	1

Чтение-запись осуществляется через аккумулятор t. ADDRESS — номер порта (0-255).

Примеры программирования на TF16

В качестве примеров программирования сначала приведем два примера: первый на Форт-ассемблере, второй на Форте. Драйвер последовательного порта RS232 написан на Форт-ассемблере (табл. 14).

Таблица 14. Драйвер последовательного порта RS232

Название фрагмента программы	Коды программы
чтение бита готовности приемника	CODE ?RS (--> 0/1) ppush -- push стека параметров rscnt1@ 2 -- чтение бита готовности returnEND-CODE
чтение байта из последовательного порта 1	CODE RS@ (--> byte) ppush rsport@ -- читаем байт rsclr1! 2 -- сбрасываем бит готовности return END-CODE
запись байта в последовательный порт 1	CODE RS! (byte -->) 1$: ppush rscnt1@ 1 -- читаем бит готовности bzp 1$ rscnt1! 1 -- сбрасываем бит готовности 2nop rsport1! -- записываем байт ppop -- pop стека параметров return END-CODE

Драйверы Flash-диска написаны на Форте. Они приводятся как пример выполнения Форт-программы (табл. 15).

Таблица 15. Драйверы Flash-диска

```
-- Работа с SPI TF16
-- Частота = 1/2 частоты процессора (12,5 МГц)
-- Передача 1 байта занимает 16 тактов

HEX

MACRO SPICSL    spics0l   END-MACRO  -- установить CS=0
MACRO SPICSH    spics0h   END-MACRO  -- установить CS=1
MACRO SPI!      spi! ppop END-MACRO  -- Запись в SPI
MACRO SPI@      ppush spi@ END-MACRO -- Чтение из SPI
MACRO DUPSPI!   spi!      END-MACRO
MACRO DROPSPI@ spi@       END-MACRO
MACRO NOP       nop       END-MACRO
MACRO 2NOP      2nop      END-MACRO

  : 8NOP 2NOP ;                    -- задержка на 8 тактов
  : 16NOP 8NOP 2NOP ;              -- задержка на 16 тактов
  : SPICS0  SPICSL 8NOP ;          -- setup time = 250ns = 8clock
  : SPICS1  8NOP SPICSH ;

-- Ожидание готовности от SPI

  : SREADY?  ( --> 0/-1, -1-OK )
    0 BEGIN
      SPICS0 57 DUPSPI! 16NOP DROP 0 DUPSPI!
      16NOP SPICS1 DROPSPI@ 80 AND IF DROP -1 EXIT THEN
    1- DUP 0= UNTIL ;

-- Запись кода

  : CodeToSPI ( номер страницы, код  --> )
    SPICS0 SPI! 2* 2* DUP ><
    16NOP  SPI! 16NOP
        SPI! 16NOP 0 SPI! 16NOP ;          -- cod,ah,al,0

-- Записать страницу 528 байт из буфера в память

  : WritePage  ( адрес буфера, номер страницы -- 0/-1, 0 - OK )
    SREADY? IFNOT 2DROP -1 EXIT THEN
    82 CodeToSPI
-- запись страницы из буфера во флэш
    108 FOR
                  DUP @ >< DUPSPI! 16NOP
                  >< SPI! 2+ 8NOP NOP
        ENDFOR
    DROP 0 SPICS1 ;

-- Читать страницу 528 байт в буфер

  : ReadPage   (адрес буфера, номер страницы -- 0/-1, 0 - OK )
    SREADY? IFNOT 2DROP -1 EXIT THEN
    52 CodeToSPI
      0 DUPSPI! 16NOP DUPSPI! 16NOP
      DUPSPI! 16NOP SPI! 16NOP             -- 0,0,0,0

-- Чтение в буфер
    0 DUPSPI! 16NOP
    108 FOR
            DROPSPI@ DUPSPI! >< 8NOP 2NOP 2NOP 2NOP
            SPI@ DUPSPI! OR SWAP 2DUP! 2+ SWAP 8NOP NOP
        ENDFOR
  2DROP 0 SPICS1 ;
```

Пример загрузки и отладки (в данном случае — запуск на исполнение) программы при помощи монитора. Программа написана в среде целевого компилятора ИнфоФорт (табл. 16).

Таблица 16. Пример работы с TF16 при помощи монитора

```
{S
 DECIMAL
 0000h 1000h TARGROM ROM    -- 0000h — начальный адрес
                               целевой памяти
                            -- 1000h — размер целевой
                               памяти
{T
ROM

«Hello, WORLD!» STRING HW

: MAIN
PAGE                        -- очистка экрана
0 0 AT-XY                   -- установили координаты курсора
HW COUNT TYPE               -- вывели строку
BEGIN                       -- бесконечный цикл
  AGAIN
;

KS!                         -- добавили контрольную сумму

' MAIN RSRUN                -- загрузили программу и выполнили
```

Сравнение производительности процессоров TF16 и i486

В таблице 17 проводится сравнение процессора TF16, реализованного на FPGA фирмы Altera, с процессором i486 фирмы Intel.

Таблица 17. Сравнение характеристик процессоров TF16 и i486

Характеристика	TF16	i486
Сложность	Около 1000 логических элементов Altera Acex EP1K30	1 200 000 транзисторов
Разрядность данных	16	32
Длина команды	• 2 байта (вызов, условный переход, арифметика) • 4 байта (длинный переход >64 кбайт)	от 1 до 17 байт
Время реакции на прерывание	От 4 до 6 тактов в зависимости от текущей команды (однотактовая, двухтактовая)	от 45 до 181 такта в зависимости от режима процессора, состояния кэша и типа обработчика прерывания

Результаты испытаний производительности процессоров

Для сравнения использовались программы, написанные для i486 на языке Турбо Паскаль 5.5 или на языке Си для компилятора DJGPP версии 2. Программа для TF16 написана на языке Форт и выполнялась на плате с процессором TF16. Программы для i486 и TF2016 прогонялись на одних и тех же данных и выдавали один и тот же результат вычислений. Время выполнения тестов для испытываемых процессоров приведено в таблице 18.

По результатам сравнения можно сделать следующий вывод: процессор TF16 разработан специально для применения в системах реального времени, с возможностью быстрого переключения контекстов в многозадачном режиме. Стековая архитектура в сочетании с продуманной системой команд обеспечивает реальное высокое быстродействие без использования конвейеров команд и кэшей.

Для удобства пользователей фирма выпустила модуль TF16, внешний вид которого приведен на рис. 16. Модуль представляет собой сборку из микросхем, перечисленных в таблице 19 и устанавливается на основную плату как мезонинный. Такая конструкция позволяет выполнить основную плату меньшего класса точности и снимает все проблемы по отладке и ремонту процессорного узла. Модуль содержит память и вспомогательный микроконтроллер, предназначенный для хранения кодов защиты как для самого модуля TF16, так и для задач пользователя, запускаемых на исполнение в данном модуле. По требованию заказчика модуль TF16 может быть выполнен с контроллером Ethernet 10/100 «на борту» и с программной поддержкой TCP/IP.

Модуль представляет собой микрокомпьютер, который может применяться для различных встроенных систем и способен заменить любые 16-разрядные встроенные микропроцессоры, например, такие как 386EX.

Таблица 19. Перечень микросхем модуля процессора TF16

Тип	Название	Корпус
Flash	K9F2808U0B-YCB0	TSOP1-48
FPGA	EP1K30TC144	QFP144
EPROM	AT17LV010-JI	DIP8
RAM	K6R4016V1B	TSOP2-44
DataFlash	AT45DB321B	32T-TSOP
Controller	ATTINY15L-1SI	SO8_ATTINY

Заключение

Заканчивая этот обзор, хочу напомнить читателям, почему эта статья называется «Стековые процессоры или Новое — это хорошо забытое новое». Технология стековых процессоров появилась в нашей стране во время перестройки. В то время это была достаточно новая микропроцессорная архитектура. Но первые отечественные стековые микропроцессоры «вымерли», едва успев родиться, после чего эта технология для многих применений была закрыта. Сказывалось не только отсутствие элементной базы, но, в основном, отсутствие специалистов. Для молодых инженеров оказалось гораздо проще получить информацию о массовых микроконтроллерах, программируемых на C/C++. С появлением технологии FPGA появилась возможность выполнять системы на кристалле и применять там те процессоры, которые наиболее полно удовлетворяют целям и задачам разрабатываемого проекта. Автор надеется, что данная статья окажется полезной при выборе направления в разработке систем на кристаллах и встроенных систем.

Таблица 18. Результаты испытаний процессоров на различных тестах

Программа	TF2016 (25 МГц)	80486DX2-50 (25 МГц)	80486DX2-50, turbo (50 МГц)
CRC-32, байтовый алгоритм: 4096 раз посчитать crc от 2048 байт, Forth/Pascal	37,9 с	58,5 с	33,1 с
CRC-32, побитовый алгоритм: 256 раз посчитать crc от 2048 байт, Forth/Pascal	18,7 с	25,9 с	11,2 с
Функция Аккерманна (3,9), Forth/C	8,3 с	32,2 с	11,4 с
Функция Аккерманна (3,9), Ассемблер	5,6 с	10,9 с	4,2 с
Быстрая сортировка, 4096 целых чисел, 256 раз, Forth/C	22,0 с	34,6 с	13,6 с
Перемножение матриц 100×100 16 раз, Forth/C	47,4 с	61,4 с	29,8 с
34-е число Фибоначчи, Forth/оптимизированный C	19,9 с	27,3 с	10,1 с
34-е число Фибоначчи, Ассемблер	9,2 с	15,6 с	5,9 с

Примечания:

1. Все цифры — точные (если явно не отмечено что-то другое, разброс наблюдался не ранее, чем во втором знаке после запятой).
2. Если сравнивать Форт и Паскаль, программы написаны на языках высокого уровня, и оба компилятора порождают код, уступающий по качеству ассемблерному коду, написанному человеком.
3. i486 находился в выгодных условиях, так как и программа, и данные находились в кэше.
4. Функция Аккерманна и числа Фибоначчи показывают преимущества вызова в TF16 в сравнении с i486.

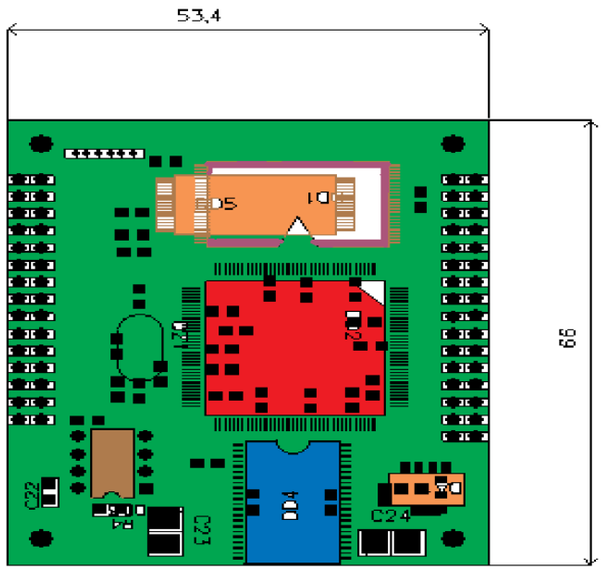

Рис. 16. Модуль TF16