Микроконтроллер для встроенного применения – NIOS. Система команд и команды, определяемые пользователем.

Часть I. Регистры и доступ к данным

Иосиф Каршенбойм

Ik@lmail.loniis.spb.su

Введение

Целью настоящей статьи является предварительное знакомство с системой команд микропроцессора Nios, описание ее отличий от команд стандартных микропроцессоров.

Рассматриваемый здесь микропроцессор относится к классу микропроцессоров для встроенного применения, то есть основная задача такого процессора — работа в режиме реального времени. Для этого микропроцессор должен быстро реагировать на прерывания, быстро переключаться с одной задачи на другую.

При реализации процессора в FPGA появляется дополнительная возможность — реализация команд и аппаратных сопроцессоров, определяемых и разрабатываемых пользователем.

В январе 2002 года был опубликован документ — описание процессора Nios в версии 2.0 [5]. Фирмой Altera были произведены важные дополнения к возможностям системы — введен режим мультимастера в шину Avalon и добавлены новые команды — команды, определяемые пользователем (заказные команды). Документация по системе команд процессора Nios 2.0 была любезно предоставлена автору фирмой «ЭФО».

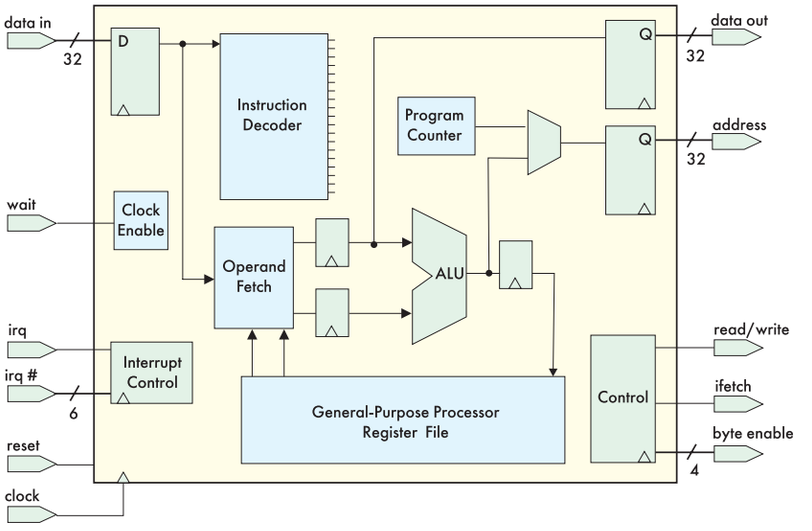

Рис. 1. Блок-схема процессора NIOS CPU-32

Описание команд, приведенное в настоящей статье, охватывает, по возможности, обе версии — Nios CPU-32 и Nios CPU-16. Но, поскольку Nios CPU-32 имеет больше команд, то таблица формата команд и кодов операций команд приведены только для 32-битной версии.

Данная статья предназначается только для ознакомления с системой команд и принципом работы микропроцессора и не может претендовать на полное рабочее руководство для программиста. Детальные описания команд процессора приведены в документации [3–5].

Развитие элементной базы происходит столь стремительно, что если в 2001 году процессором среднего быстродействия считались процессоры с тактовой частотой до 50 МГц, то теперь, с появлением на рынке микросхем серии STRATIX, в эту категорию попадают изделия с частотой до 300 МГц. А это уже серьезный конкурент для многих «средних» DSP. Более того, изменения, внесенные в шину Avalon, позволяют теперь реализовать режим мультимастера, что позволяет еще более увеличить производительность всей системы с процессором Nios [7].

Краткий обзор CPU Nios

Процессор Nios, является RISC-процессором с конвейерной обработкой команд, большинство команд которого выполняется в единственном цикле синхрочастоты [1–2, 6]. Система команд Nios ориентирована на встроенные прикладные программы. Nios CPU в вариантах применения на 16 бит и 32 бит, имеет размеры слова соответственно 16 бит и 32 бит.

В Nios термин «байтное слово» относится к 8-битному слову, «слово половинной разрядности» относится к 16-битному слову, а «слово» — к 32-битному слову. Семейство программного ядра процессора Nios включает 32- и 16-битные варианты архитектуры (далее Nios CPU-32 и Nios CPU-16) (см. табл. 1).

Таблица 1. Архитектура процессора Nios

Компоненты процессора	Nios CPU-32	Nios CPU-16
Разрядность шины данных	32	16
Разрядность ALU	32	16
Разрядность внутренних регистров	32	16
Разрядность шины адреса	32	16
Разрядность команды	16	16

Для выполнения задач реального времени процессор Nios имеет развитую систему прерываний и программных исключений.

Nios CPU — это конвейерная RISC-архитектура. Конвейер команд позволяет увеличить быстродействие процессора при выполнении кода программы, не имеющего ветвлений. Если же необходимо выполнить переход на другой адрес программы, то необходимо произвести перезагрузку конвейера команд (при этом образуется «слот задержки перехода»). То есть команда, следующая немедленно после JMP и CALL, выполняется после JMP и CALL, но перед выполнением команды перехода в новый адрес. Чтобы повысить быстродействие при коротких переходах, есть пять условно-выполняемых команд SKIP (SKPs, SKP0, SKP1, SKPRz, и SKPRnz). Каждая из этих команд позволяет выполнить при определенном условии следующую за командой SKIP команду или следующая команда пропускается. Поскольку процессор Nios имеет конвейер команд, то пропуск (невыполнение) команды выполняется за один такт синхрочастоты, в то время как перезагрузка конвейера команд при выполнении команды перехода типа JMP или CALL требует несколько циклов синхрочастоты. Поэтому применение команд SKIP для коротких переходов повышает скорость работы.

Процессор Nios имеет две команды для программной обработки исключения: TRAP и TRET. При помощи этих команд осуществляется эффективная отладка программ. Эти команды не приводят к исполнению следующей за ними команды в конвейере команд.

В процессор Nios в версии 2.0 введены дополнительные команды — команды, определяемые пользователем (заказные команды).

Nios CPU поставляется с транслятором GNUPro и отладчиком от Cygnus, что представляет собой промышленный стандарт по C/C++, с открытыми исходными материалами, компоновщиком транслятора и комплектом отладочных программ. Комплект инструментов GNUPro включает C/C ++ транслятор, макроассемблер, компоновщик, отладчик, утилиты и библиотеки.

Система команд процессора Nios ориентирована на то, чтобы эффективно поддержать программы, откомпилированные из исходных программ, написанных на C и C++. Поэтому набор команд включает в себя стандартный набор арифметических и логических операций, команды поддержки для операций над битами, извлечения байта, передачи данных, модификации потока управления, и небольшого набора условно-выполняемых команд, которые могут быть полезны в устранении условных ветвлений на короткие переходы. В процессор Nios в версии 2.0 введены команды, определяемые и разрабатываемые пользователем.

Краткий обзор регистров

Архитектура Nios CPU имеет большой файл регистров общего назначения, несколько регистров машинного управления, счетчик программы и регистр K, используемый для выполнения префиксов команд.

Регистры общего назначения

Регистры общего назначения имеют разрядность 32 бит в Nios CPU-32 и 16 бит в Nios CPU-16. Размер файлового регистра может быть переконфигурирован пользователем на этапе формирования ядра процессора и периферии и содержать 128, 256 или 512 регистров. Программное обеспечение может обращаться к регистрам через двигающееся окно длиной в 32 регистра, которое перемещается с шагом в 16 регистров. Это окно может передвигаться по всему регистровому файлу и обеспечивает доступ к подмножеству регистров файла.

Окно разделено на четыре четных раздела как показано в таблице 2. Самые нижние восемь регистров (%r0–%r7) — глобальные регистры, также называемые %g0–%g7. Глобальные регистры не изменяются с движением окна, и остаются доступными как %g0–%g7. Верхние 24 регистра (%r8–%r31) в регистровом файле доступны через текущее окно.

Таблица 2. Группы регистров

In регистры	%r24 – %r31 или %i0 – %i7
Local регистры	%r16 – %r23 или %L0 – %L7
Out регистры	%r8 – %r15 или %o0 – %o7
Global регистры	%r0 – %r7 или %g0 – %g7

Восемь верхних регистров (%i0–%i7) называются «входные» («in») регистры, следующие восемь (%L0–%L7) — «локальные» регистры, остальные восемь (%o0–%o7) -— «выходные» («out») регистры. Когда окно регистра перемещается вниз на 16 регистров (так как это делается для команды SAVE), регистры «out» становятся регистрами «in» в новой позиции окна. Кроме того, «локальные» и «in» регистры последней позиции окна становятся недоступными (табл. 3).

Регистр K

Регистр K — 11-битовый префикс, который всегда устанавливается в 0 каждой командой, кроме PFX. Системы команд PFX устанавливают регистр K непосредственно из поля команды IMM11. Регистр K содержит значение, отличное от нуля только для команды, непосредственно следующей за PFX. Команда PFX отключает прерывания для одного цикла синхрочастоты, то же происходит и с двумя командами последовательности PFX (PFX немедленно после команды SKPx или IFx) в «атомных операциях» CPU.

Кроме того, пары команды последовательности PFX пропускаются при выполнении команды SKP — команды условного выражения типа SKIP.

Программное обеспечение непосредственно не обращается к регистру K, но использует его косвенно.

Команда MOVI, например, передает все 11 бит регистра K в биты 15–5 регистра адресата. Эта операция чтения K выдаст результат, отличный от нуля только тогда, когда предыдущая команда — PFX.

Таблица 3. Программная модель

		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IN	%i7 %r31	SAVED return-address																															
	%i6 %r30	%fp-frame pointer																															
	%i5 %r29																																
	%i4 %r28																																
	%i3 %r27																																
	%i2 %r26																																
	%i1 %r25																																
	%i0 %r24																																
LOCAL	%L7 %r23																																
	%L6 %r22																																
	%L5 %r21																																
	%L4 %r20																																
	%L3 %r19	Base-pointer 3 for STP/LDP (or geenral-purpose local)																															
	%L2 %r18	Base-pointer 2 for STP/LDP (or geenral-purpose local)																															
	%L1 %r17	Base-pointer 1 for STP/LDP (or geenral-purpose local)																															
	%L0 %r16	Base-pointer 0 for STP/LDP (or geenral-purpose local)																															
OUT	%o7 %r15	current return-address																															
	%o6 %r14	%sp-Stack Pointer																															
	%o5 %r13																																
	%o4 %r12																																
	%o3 %r11																																
	%o2 %r10																																
	%o1 %r9																																
	%o0 %r8																																
GLOBAL	%g7 %r7																																
	%g6 %r6																																
	%g5 %r5																																
	%g4 %r4																																
	%g3 %r3																																
	%g2 %r2																																
	%g1 %r1																																
	%g0 %r0																																
		31																					10										0
																							K REGISTER										
		PC																															
%ctl9	SET_IE	Any write (WRCTL) operation to this register sets STATUS [15] (IE)=1. Result of any read operation (RDCTL) is undefined.																															
%ctl8	SET_IE	Any write (WRCTL) operation to this register sets STATUS [15] (IE)=1. Result of any read operation (RDCTL) is undefined.																															
%ctl7	—	— reserved —																															
%ctl6	—	— reserved —																															
%ctl5	—	— reserved —																															
%ctl4	—	— reserved —																															
%ctl3	—	— reserved —																															
%ctl2	WVALID																						HI_LIMIT							LO_LIMIT		
%ctl1	ISTATUS																	Saved Status														
%ctl0	STATUS																	IE	IP RI						CWP					N	V	Z	C
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Счетчик команд

Счетчик команд — регистр PC — содержит адрес байта команды, выполняющейся в настоящее время. Так как все команды должны быть выровнены до слова половинной разрядности, значение бита младшего разряда PC всегда равно 0.

PC инкрементируется на два (PC <– PC + 2) после каждой команды, если PC не устанавливается непосредственно. Команды BR, BSR, CALL и JMP изменяют PC непосредственно. PC имеет разрядность 33 бит в Nios CPU-32 и 17 бит в Nios CPU-16.

Регистры управления

Есть пять определенных регистров управления, которые адресуются независимо от регистров общего назначения. Команды RDCTL и WRCTL — единственные команды, которые могут читать или писать в регистры управления (значение %ctl0 не связано с %g0).

STATUS (%ctl0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IE	IPRI						CWP					N	V	Z	C

Разрешение прерывания (IE)

IE — бит разрешения прерывания. Когда IE=1, все внешние прерывания и внутренние исключения разрешены. IE=0 отключает внешние прерывания и исключения.

Программные команды TRAP будут все же нормально выполняться, даже когда IE=0.

Обратите внимание, что IE может быть установлен непосредственно при записи в SET_IE (%ctl9) и CLR_IE (%ctl8) в регистр управления, и это не будет воздействовать на остальную часть регистра STATUS. Когда CPU находится в состоянии сброса, IE будет установлен в 0 (заблокированные прерывания).

Приоритет прерываний (IPRI)

IPRI содержит приоритет выполняющегося прерывания. Когда исключение обработано, значение IPRI установлено на номер исключения.

Для внешних аппаратных прерываний значение IPRI соответствует 6-битовому номеру аппаратного прерывания. Для команд TRAP поле IPRI устанавливается в соответствии со значением поля команды IMM6. Для внутренних исключений IPRI устанавливается от 6-битового номера исключения. Аппаратное прерывание не будет обработано, если его внутренний номер больше или равен IPRI или IE=0. Команда TRAP обрабатывается всегда. Когда CPU сброшен, IPRI установлен на 63 (самый низкий приоритет).

IPRI запрещает прерывания выше некоторого номера. Например, если IPRI — 3, то прерывания 0, 1 и 2 будут обработаны, но все другие (прерывания 3–63) будут заблокированы.

Текущий указатель окна (CWP)

CWP указывает на базовый адрес регистра для окна, перемещающегося в регистровом файле общего назначения. Инкрементирование CWP перемещает окно на 16 регистров вверх. Декрементирование CWP перемещает окно вниз на 16 регистров. CWP декрементируется командами SAVE и инкрементируется командами RESTORE.

Только специализированное системное программное обеспечение, такое, как «средство для управления окном регистра» должно непосредственно записывать значения CWP через WRCTL. Обычное программное обеспечение будет изменять CWP, используя команды SAVE и RESTORE.

Когда CPU сброшен, CWP будет установлен на самое большое допустимое значение — HI_LIMIT.

После того, как произведен сброс, регистр WVALID (%ct12) установлен в 0x01C1, то есть LO_LIMIT = 1 и HI_LIMIT = 14 (см. раздел «WVALID (%ctl2)» для получения дополнительной информации).

Флажки состояния

Некоторые команды изменяют флажки состояния. Эти флажки — четыре младших бита регистра состояния (табл. 4).

Таблица 4. Флаги состояния

Флаг	Бит	Результат
N	3	Знак результата или старший бит
V	2	Арифметическое переполнение, если бит 31 из 32-битового результата отличается от знака результата, вычисленного с неограниченной точностью
Z	1	Результат равен нулю
C	0	Перенос при сложении, заем при вычитании

ISTATUS (%ctl1)

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

ISTATUS — это сохраненная копия регистра STATUS. Когда исключение обработано, значение регистра STATUS будет скопировано в регистр ISTATUS. Это действие позволяет восстанавливать значение регистра STATUS перед возвращением управления прерванной программе. При возвращении из TRAP команда TRET автоматически копирует регистр ISTATUS в регистр STATUS. Когда исключение обрабатывается, прерывания заблокированы (IE=0) . Перед перепредоставлением прерываний драйвер исключения должен сохранить значение регистра ISTATUS. Когда CPU сброшен, ISTATUS установлен в 0.

WVALID (%ctl2)

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

WVALID содержит два значения: HI_LIMIT и LOW_LIMIT. Когда выполняются декременты CWP по команде SAVE от значения LOW_LIMIT до LOW_LIMIT-1, происходит выход за нижний предел окна регистра и генерируется исключение #1. Когда команда RESTORE инкрементирует CWP от HI_LIMIT до HI_LIMIT + 1, происходит переполнение окна регистра и генерируется исключение #2. Регистр WVALID имеет перестраиваемую конфигурацию и может быть доступен только для чтения или для чтения и записи. Когда CPU сброшен, LO_LIMIT устанавливается в 1, а HI_LIMIT равен самому высокому допустимому указателю окна — (размер файла регистра / 16) — 2.

CLR_IE (%ctl8)

Любая операция WRCTL с регистром CLR_IE сбрасывает бит IE в регистре STATUS (IE <– 0) и значение WRCTL игнорируется. Операция RDCTL над CLR_IE приводит к неопределенному результату.

SET_IE (%ctl9)

Любая операция WRCTL с регистром SET_IE устанавливает бит IE в регистре STATUS (IE <– 1) и значение WRCTL игнорируется. Операция RDCTL над SET_IE приводит к неопределенному результату.

Доступ к памяти

Процессор Nios — организован как little-endian. Память данных должна находиться в непрерывном порядке следования слов. Если физическое устройство памяти более узко, чем размер рабочего слова, то шина данных должна осуществить установление размеров динамическим образом, чтобы моделировать для Nios CPU данные полной разрядности. Периферийные устройства представляют свои регистры как native-word, то есть младшие разряды подключаются к младшим разрядам шины, и если их регистры меньше, чем рабочее слово, то остальные разряды дополняются нулями в старших битах. В таблице 5 показано содержимое типового поля памяти, а в таблице 6 показаны регистры, подключаемые как «native-word» к Nios CPU-32.

Таблица 5. Поле памяти программ и данных для Nios CPU-32 с адреса 0x0100

Адрес	Содержание поля памяти			
	31 24	23 16	15 8	7 0
0x0100	Байт 3	Байт 2	Байт 1	Байт 0
0x0104	Байт 7	Байт 6	Байт 5	Байт 4
0x0108	Байт 11	Байт 10	Байт 9	Байт 8
0x010C	Байт 15	Байт 14	Байт 13	Байт 12

Таблица 6. Периферия разрядностью N-бит для Nios CPU-32 с адреса 0x0100

Адрес	Содержание регистров периферии	
	31 N	N-1 0
0x0100	Заполнение нулями	Регистр 0
0x0104	Заполнение нулями	Регистр 1
0x0108	Заполнение нулями	Регистр 2
0x010C	Заполнение нулями	Регистр 3

Чтение из памяти или периферийных устройств

Nios CPU может только выполнять выровненный доступ к памяти. 32-битная операция чтения может читать только полное слово, начинающееся с адреса байта, который является множителем 4. 16-битная операция чтения может читать только слово половинной разрядности, начинающееся с адреса байта, который является множителем 2. Команды, которые читают из памяти всегда, устанавливают младший бит (для Nios CPU-16) или два младших бита (для Nios CPU-32) адреса как в 0.

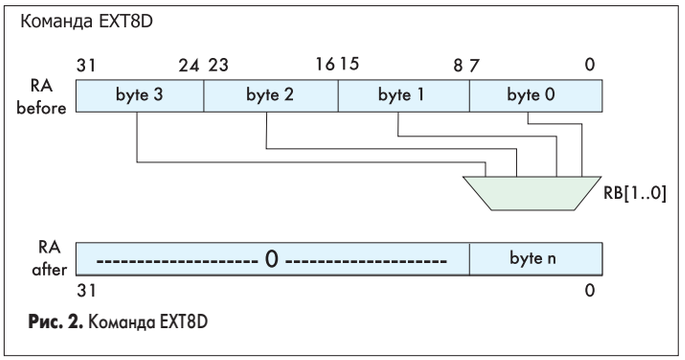

Рис. 2. Команда EXT8D

Пример кода 1. Чтение одиночного байта памяти

```
Contents of memory:

;                   0       1       2       3
;  0x00001200       0x46    0x49    0x53    0x48

;  Instructions executed on  a 32-bit Nios CPU

                            ;   Let's assume %04 contains the address x00001202

LD %g3, [%04]               ;   %g3 gets the contents  of address 0x1200,
                            ;   so %g3 contains 0x48534946
EXT8D %g3, %04              ;   %g3 gets replaced with byte 2 from %g3,
                            ;   so %g3 contains 0x00000053
```

Рис. 3

Команды для чтения отдельных байтов и слов половинной разрядности

Самая простая команда, которая читает данные из памяти — команда LD. Типичный пример этой команды — LD %g3, [%o4]. Первый операнд регистра, %g3, является регистром адресата, куда данные будут загружены. Второй операнд регистра определяет регистр, содержащий адрес, из которого будет произведено чтение. Этот адрес будет выровнен к самому близкому слову половинной разрядности (для Nios CPU-16) или к слову (для Nios CPU-32), причем значение самого младшего бита (для Nios CPU-16) или двух бит адреса (для Nios CPU-32) будут обработаны, как если бы они равнялись 0.

Весьма часто, однако, программное обеспечение должно читать данные меньшие, чем рабочее слово данных. Nios CPU обеспечивает команды для распаковки индивидуальных байтов (для Nios CPU-16 и для Nios CPU-32) и словами половинной разрядности (для Nios CPU-32) от рабочего слова данных. Команда EXT8D используется для распаковки байта, а EXT16D — для распаковки слова. Типичный пример команды EXT8D — EXT8D %g3, %o4. Команда EXT8D использует самый младший бит (для Nios CPU-16) или два бита (для Nios CPU-32) второго операнда регистра, чтобы извлечь байт из первого операнда регистра и заменить содержимое первого операнда регистра тем байтом.

На рис. 3 показано, как прочитать отдельный байт из памяти, даже если адрес байта не приведен к native-word.

Запись в память или периферийные устройства

Процессор Nios может выполнять выровненную запись в память слов, имеющих разрядность байта, слова половинной разрядности или слова. Слово (для Nios CPU-32) может быть записано в любой адрес, кратный 4, одной командой. Слово половинной разрядности может быть записано в любой адрес, кратный 2, одной командой (для Nios CPU-16) или двумя командами (для Nios CPU-32). Байт может быть записан в любой адрес двумя командами.

В Nios CPU-32 самый младший байт регистра может быть записан только по адресу, кратному 4; следующий байт может быть записан только по адресу, кратному 4, плюс 1, и т. д.

Точно так же для Nios CPU-16, младший байт регистра может быть записан только по четному адресу, а старший байт может быть записан только по нечетному адресу.

Nios CPU-32 может также записать младшее слово половинной разрядности в регистр по адресу, кратному 4, а старшее слово половинной разрядности регистра — по адресу, кратному 4, плюс 2.

Команда ST пишет полное слово рабочей разрядности, взятое из любого регистра с выравниванием типа «native». Команды ST8D и ST16D (только для Nios CPU-32) пишут байт и слово половинной разрядности, соответственно, с ограничениями выравнивания, описанными выше, от регистра %r0.

Для программного обеспечения часто бывает необходимо записать специфический байт или слово половинной разрядности по произвольно расположенному адресу в памяти. Позиция в пределах исходного регистра может не соответствовать расположению в памяти, в которую будет производиться запись.

Команды FILL8 и FILL16 (только для Nios CPU-32) будут брать самый младший байт или слово половинной разрядности регистра соответственно, и копировать это в регистр %r0.

На рис. 4 показано, как записать отдельный байт в память, даже если адрес байта не приведен к native.

Пример кода 2. Запись в память одного байта
– Адрес не выровнен к ширине слова.

```
Содержание памяти до операции:

                            0       1       2       3
0x00001200                  0x46    0x49    0x53    0x48

;     let`s assume %04 contains the address x00001203
;     and that %g3 contains the value 0x000000BC
FILL8 %r0, %g3      ;   (First operand can only be %r0)
                    ;   replicate low byte of %g3 across %r0
                    ;   so %r0 contains 0xBCBCBCBC
ST8D [%04], %r0     ;   (Second operand can only be %r0)
                    ;   Stores the 3rd byte of %r0 to address 0x1203

Содержание памяти после операции

                            0       1       2       3
0x00001200                  0x46    0x49    0x53    0xBC
```

Рис. 4

Способы адресации

Nios CPU поддерживает 5 способов адресации:

- 5/16-битная прямая адресация;
- косвенная адресация полной разрядности;
- косвенная адресация частичной разрядности;
- косвенная адресация полной разрядности со смещением;
- косвенная адресация частичной разрядности со смещением.

5/16-битная прямая адресация

Множество арифметических и логических команд используют непосредственное 5-битное число в качестве операнда. На рис. 5 показана команда ADDI с двумя операндами: регистром и 5-битным числом. 5-битное число — это константа от 0 до 31. Чтобы определить постоянное значение, требуется от 6 до 16 бит (число от 32 до 65535), 11 бит регистра K могут быть установлены с помощью команды PFX. Это значение связано с 5-битным числом. Команда PFX должна использоваться непосредственно перед командой, которую она изменяет.

Пример кода 3. Применение команды ADDI с PFX и без PFX

```
                            ;  Assume %g3 contains the value 0x00000041
ADDI %g3,5                  ;  Add 5 to %g3
                            ;  so %g3 now contains 0x00000046
PFX %hi (0x1234)            ;  Load K with upper 11 bits of 0x1234
ADDI %g3, %1o (0x1234)      ;  Add low 5 bits of 0x1234 to %g3
                            ;  so the K register contained 0x091
                            ;  and the immediate operand of the ADDI
                            ;  instruction contained 0x14, which
                            ;  concatenated together make 0x00001234
                            ;  Now %g3 contains 0x0000127A
```

Рис. 5

Для преобразования 16-битной константы в значение PFX и 5-битное число ассемблер использует операторы %hi () и %lo (). Оператор %hi (x) получает от константы x 11 бит (с 5 по 15 биты), а %lo (x) — 5 бит (с 0 по 4 биты).

Помимо арифметических и логических команд есть еще несколько команд, которые используют режим прямой адресации различной разрядности без использования регистра K. В таблице 7 приведены команды, использующие 5/16-битные числа. Команды, помеченные звездочкой, — AND, ANDN, OR и XOR — могут только использовать 16-битные значения PFX. Эти команды действуют на два операнда регистра, если им не предшествовала команда PFX.

Таблица 7. Команды, использующие 5/16-битную прямую адресацию

ADDI	AND*	ANDN*	ASRI
CMPI	LSLI	LSRI	MOVI
MOVHI	OR*	SUBI	XOR*

Косвенная адресация полной разрядности

Команды LD и ST могут загружать и сохранять, соответственно, полное рабочее слово в или из регистра, используя другой регистр. Чтобы определить адрес, см. таблицу 8.

Сначала адрес должен быть приведен к рабочему слову так, как описано выше, а регистр K должен быть обработан как смещение со знаком, разрядностью в рабочее слово.

Таблица 8. Команды, использующие косвенную адресацию полной разрядности

Команда	Адресный регистр	Регистр данных
LD	Любой регистр	Любой регистр
ST	Любой регистр	Любой регистр

Косвенная адресация частичной разрядности

Среди команд Nios нет команд, которые читают слово частичной разрядности. Чтобы прочитать частичное слово, вы должны комбинировать (объединить) команду косвенного чтения полной разрядности с командой извлечения — EXT8d, EXT8s, EXT16d (только для Nios CPU-32) или EXT16S (только для Nios CPU-32).

Есть несколько команд, которые могут записывать частичное слово. Каждая из этих команд имеет статический и динамический вариант. Позиция и в пределах исходного регистра, и в рабочем слове памяти определена младшими битами регистра адресации. В случае статического варианта позиция и в пределах исходного регистра, и рабочего слова памяти определена как 1- или 2-битный непосредственный операнд в команде. Как и в случае косвенной адресации через регистр с полной разрядностью, регистр K трактуется как знаковое смещение в рабочем слове при выравнивании адреса по рабочему слову.

Таблица 9. Команды, использующие косвенную адресацию частичной разрядности

Команда	Адресный регистр	Регистр данных	Выбор байта или половины слова
ST8S	Любой регистр	%r0	Непосредственный
ST16S	Любой регистр	%r0	Непосредственный
ST8D	Любой регистр	%r0	Младшие биты регистра адреса
ST16D	Любой регистр	%r0	Младшие биты регистра адреса

Все команды записи данных частичной разрядности через косвенный регистр используют %r0 в качестве источника данных. Эти команды удобно использовать вместе с командами FILL8 или FILL16 (только для Nios CPU-32) (см. табл. 9).

Косвенная адресация полной разрядности со смещением

Команды LDP, LDS, STP и STS могут загружать или записывать полное рабочее слово в или из регистра, используя другой регистр для определения адреса и число, чтобы определить смещение в рабочих словах от этого адреса.

В отличие от команд LD и ST, которые для определения адреса памяти могут использовать любой регистр, эти команды могут использовать только специальные регистры для формирования адреса. Команды LDP и STP могут использовать только регистры %L0, %L1, %L2 или %L3 для формирования адреса. Команды LDS и STS могут только использовать указатель стека — регистр %sp (эквивалент %o6) в качестве регистра адреса. Каждая из этих команд берет непосредственное индексное значение со знаком, которое определяет смещение в рабочих словах от выровненного адреса (см. табл. 10).

Таблица 10. Команды, использующие косвенную адресацию полной разрядности со смещением

Команда	Адресный регистр	Регистр данных	Диапазон смещения без PFX
LDP	%L0, %L1, %L2, %L3	Любой регистр	0...124 байт
LDS	%sp	Любой регистр	0...1020 байт
STP	%L0, %L1, %L2, %L3	Любой регистр	0...124 байт
STS	%sp	Любой регистр	0...1020 байт

Косвенная адресация частичной разрядности со смещением

Среди команд Nios нет команд, которые читают частичное слово из памяти. Чтобы читать частичное слово, вы должны комбинировать (объединить), индексную косвенную команду чтения полной разрядности с командой извлечения — EXT8d, EXT8s, EXT16d (только для Nios CPU-32) или EXT16S (только для Nios CPU-32). STS8S и STS16S (только для Nios CPU-32) используют число, чтобы определить байт или смещение слова половинной разрядности, соответственно, от указателя стека, чтобы записать, соответственно, выровненное слово частичной разрядности из исходного регистра %r0 (см. табл. 11).

Таблица 11. Команды, использующие косвенную адресацию частичной разрядности со смещением

Команда	Адресный регистр	Регистр данных	Выбор байта или половины слова	Диапазон
ST8S	%sp	%r0	непосредственный	0...1023 байт
ST16S	%sp	%r0	непосредственный	0...511 слов половинной разрядности

Есть команды, которые могут использовать только указатель стека — регистр %sp (эквивалент %o6) в качестве регистра адреса и могут использовать только регистр %r0 (эквивалент %g0, но в ассемблере должен называться %r0) в качестве регистра данных. Эти команды удобно использовать с командами FILL8 или FILL16 (только для Nios CPU-32).

Литература

1. Altera™. Nios Soft Core Embedded Processor, data sheet June 2000, ver. 1.
2. Altera™. Nios_2_0_CPU_datasheet.
3. Altera™. Nios Programmer's Reference Manual Version 1.1. March 2001.
4. Altera™. Nios 32-Bit Programmer's Reference Manual. January 2002. Version 2.0.
5. Altera™. Nios 16-Bit Programmer's Reference Manual. January 2002. Version 2.0.
6. Altera™. Nios_tutorial.pdf.
7. Altera™. Simultaneous Multi-Mastering with the Avalon Bus.

Окончание следует