История изменений

Отличия от версии 1.0.0

1. **Добавлена команда CALRcc.**
2.

Техническое описание на
32х-разрядный конвейерный RISC процессор rz80

Общее описание

При проектировании данного микропроцессора ставилась задача разработки конвейеризированного процессора с выполнением одной команды за такт Гарвардской архитектуры, но с учетом минимального программно-аппаратного эмулирования микропроцессора z80. Типы внешних данных – 8, 16, 32 разряда со знаком и без знака. Длина команды – фиксированная в 32 разряда. Шины данных и команд – раздельные. Для эмуляции микропроцессора z80 были реализованы все его флаги, а также введены некоторые специфические команды и добавлены дополнительные 32 регистра.

Блок схема подключения процессорного ядра представлена на рисунке 2. Через интерфейс JTAG осуществляется связь программы-отладчика на ПК с внутрисхемным отладчиком ICD (In Circuit Debugger). Использование внутрисхемного отладчика позволяет отлаживать программное обеспечение непосредственно в аппаратуре. Через шину AHB к процессору могут быть подключены различные периферийные блоки, а также мост для подключения внешней 8 разрядной памяти и устройств.

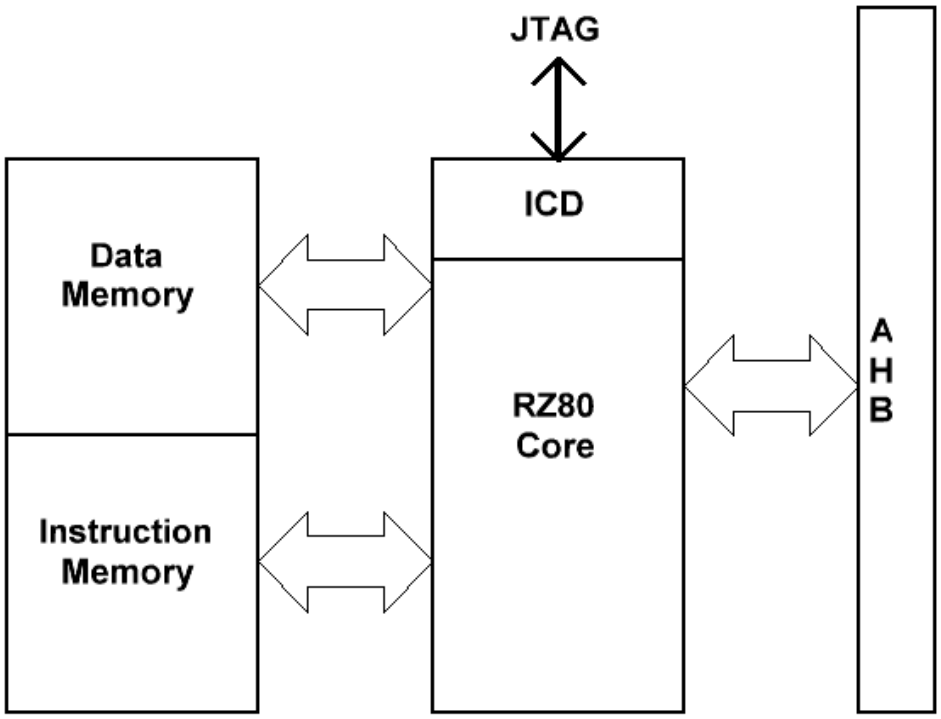


Рис.2

Описание внешних выводов процессора

SCLK – системная частота (рабочий фронт - положительный);
PRES – сброс процессора (активный высокий уровень);
ICSTALL – останов процессора при работе с кеш команд (без кеш должен быть '0');
TCK, TDI, TMS – входы JTAG;
TRSTN – сброс JTAG (активный низкий уровень);
TDO – выход данных JTAG;
TDOE – разрешение активного уровня выхода данных JTAG (активный высокий уровень);
HCLK, HGRANT, HRDATA, HREADY, HRESETn, HRESP, HADDR, HBURST, HBUSREQ, HLOCK, HPROT, HSIZE, HTRANS,HWDATA, HWRITE – выводы системной шины AMBA AHB v2.0;

Регистры

В таблице 1 показаны внутренние 32-х разрядные регистры (register file) микропроцессора rz80.

Табл. 1

Адрес регистра	Название регистра
x0	Константа 0 (R0)
x1	Указатель стека (SP)
x2	Регистр 2 (R2)
x3	Регистр 3 (R3)
x4	Регистр 4 (R4)
x5	Регистр 5 (R5)
x6	Регистр 6 (R6)
x7	Регистр 7 (R7)
x8	Регистр 8 (R8)
x9	Регистр 9 (R9)
xA	Регистр 10 (R10)
xB	Регистр 11 (R11)
xC	Регистр 12 (R12)
xD	Регистр 13 (R13)
xE	Регистр 14 (R14)
x1C	Регистр 28 (R28)
x1D	Регистр состояния (SR)
x1E	Регистр прерываний (QR)
x1F	Регистр флагов (FR)
x20	Регистр 32 (R32)
x3F	Регистр 64 (R64)

В командах обработки данных регистр R0 не доступен по записи. Это используется в командах сравнения (CMP) при указании в качестве регистра-приемника R0 (Rd = R0). В этом случае изменяется только код условий, входящий в состав регистра флагов (если формат команды RRR или RRI и разрешение установки флагов S равен '1'). При чтении R0 в командах обработки и пересылки данных подставляется константа ’0’. 32 старших регистра R32..R64 используются для хранения регистров эмулируемого процессора (z80). Микропроцессор также содержит следующие регистры: указатель на стек данных SP (stack pointer), регистр прерываний (QR), регистр состояния SR (status register), регистр флагов FR (flags register).

Указатель на стек используется для хранения промежуточных данных, которые не поместились в регистровом файле, в памяти. В качестве стека возвратов используются 16 специальных регистров, не видимых программой.

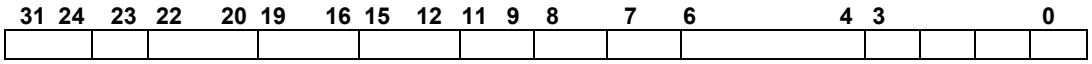
Регистр состояния (Рис. 3) служит для хранения кода условий АЛУ (см. Табл.4).

Регистр прерываний (Рис. 3) служит для разрешения прерываний и хранения флагов. Нулевой разряд GIE (Global Interrupt Enable / Глобальное разрешение прерываний) разрешает (лог.’1’) или запрещает (лог.’0’) все прерывания (кроме NMI), установленные в разрядах с 1-го по 15-й (IE1…IE15, номер разряда соответствует номеру прерывания). С приходом положительного сигнала на входы аппаратных прерываний (INTR1… INTR15), соответствующие флаги IF1…IF15 устанавливаются в ‘1’, а разряд GIE сбрасывается в ‘0’, запрещая последующие прерывания. По команде возврата из прерывания RETI разряд GIE возвращается в ‘1’. Наивысший приоритет имеет первое прерывание (не считая NMI). Прерывание можно вызвать также из программы исполнением команды TRAP с соответствующим вектором прерывания (например TRAP 1, см. Табл.5). В этом случае прерывания ведут себя как обычные подпрограммы (не влияющие на GIE и возвратом по RET).

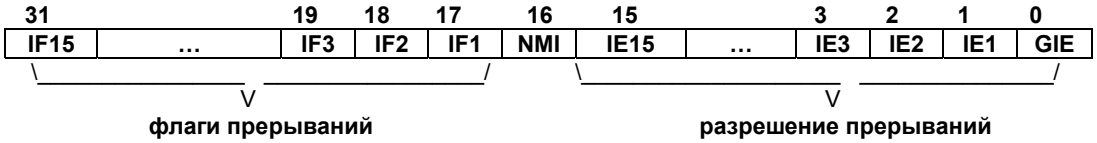
По сигналу RESET разряды всех регистров обнуляются.

Рис. 3

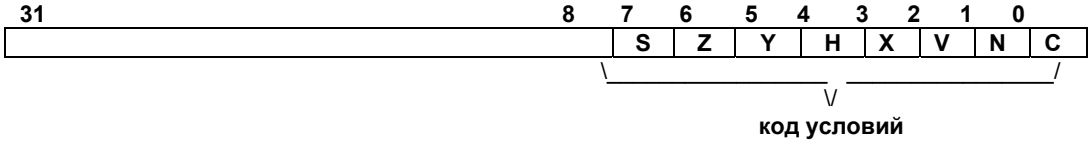
Регистр состояния (SR)



Регистр прерываний (QR)



Регистр флагов (FR)



Стек возвратов

В процессоре применен аппаратный стек возвратов. Он состоит из 16-ти специализированных регистров, напрямую недоступных со стороны программы. В командах CALL происходит занесение состояния счетчика программ +8 в стек возвратов и извлечение его в командах RET. Для возврата из прерываний также используется стек возвратов. Глубина стека составляет 16 (число регистров), поэтому программист должен отслеживать, чтобы количество вложенных команд CALL вместе с прерыванием не превышало это число.

Система команд

В микропроцессоре rz80 используются следующие форматы команд:

Таблица 1

31 28	27 24	23 22 21	16	15 14 13	8	7	6	5 0	формат
0000	0								NOP
0000	Rd	Ra	00	Rb		S	0	Opalu	RRR
0000	X	X	01	X		X	0	X	Reserv
0000	X	X	10	X		X	0	X	Reserv
0000	X	X	11	X		X	0	X	Reserv
0000	Rd	Ra	I8			S	1	Opalu	RRI
0001	Rd	(Ra)	sD16						LB
0010	Rd	(Ra)	sD16						LH
0011	Rd	(Ra)	sD16						LW
0100	Rd	(Ra)	sD16						SB
0101	Rd	(Ra)	sD16						SH
0110	Rd	(Ra)	sD16						SW
0111	Rd	(Ra)	sD16						LBU
1000	Rd	(Ra)	sD16						LHU
1001	0000	X							HALT
1001	0001	X				Intr			IRQ/TRAP
1001	0010	X				N			HBRK
1001	0011	X				N			SBRK
1001	0100	X							RETN
1001	0101	X							RETI
1001	0110	X							RETB
1001	0111	I24							IMM
1001	1XXX	X							Reserv
1010	Cond	00	Ra	X					JRcc
1010	Cond	01	Ra	X					CALRcc
1010	X	1X	X	X					Reserv
1011	Cond	sD24							BRcc
1100	Cond	sD24							CALLcc
1101	Cond	X							RETcc
1110	X								Reserv
1111	X								Reserv

X – Резерв (в данной версии должны быть нулями).
I8, I24 – Непосредственная константа.
sD16, sD24 – Знаковое смещение.
S – разрешение установки флагов условий (set condition flags).

Поле “cond” используется для условий перехода в командах ветвления. Эти условия перечислены в таблице 4.

Таблица 4

Cond	Выполняемая функция		Примечание
x0	C = 1	CS / LTU	Перенос / Результат меньше 0 (беззн.)
x1	C = 0	CC / GEU	Нет переноса / Рез. > или = 0 (беззн.)
x2	Z = 1	EQ	Результат равен 0
x3	Z = 0	NE	Результат не равен 0
x4	N xor V	LT	Результат меньше 0
x5	not (N xor V)	GE	Результат больше или равен 0
x6	V = 1	VS	Переполнение
x7	V = 0	VC	Нет переполнения
x8	Z or (N xor V)	LE	Результат меньше или равен 0
x9	not (Z or (N xor V))	GT	Результат больше 0
xA	not (Z or C)	GTU	Результат больше 0 (беззнаковый)
xB	Z or C	LEU	Результат меньше или равен 0 (беззн.)
xC	N = 1	NEG	Результат отрицательный
xD	N = 0	POS	Результат положительный
xE	Резерв		
xF	Always	UC	Безусловный переход

8 младших разрядов в командах прерывания (Intr[7..0]) используются для указания адреса прерывания. Для аппаратно-программных прерываний в адресном пространстве микропроцессора зарезервировано 32 ячейки памяти. Распределение адресов прерываний показано в таблице 5.

Таблица 5

Address	Прерывания	Примечание
x0	Reset	Прерывание по Reset
x4	IRQ 1	Внешнее прерывание 1
x8	IRQ 2	Внешнее прерывание 2
…	…	…
…	…	…
x38	IRQ 14	Внешнее прерывание 14
x3C	IRQ 15	Внешнее прерывание 15
x40	NMI	Не маскируемое прерывание
x44	TRAP 17	Внутреннее прерывание 17
…		
x7C	TRAP 31	Внутреннее прерывание 31

Первые 17 прерываний являются внешними. Их можно вызвать по внешним выводам микропроцессора. Наивысший приоритет у внешних прерываний имеет прерывание по “Reset” (x0). Внутренние прерывания TRAP (от x44 до x7C) не имеют приоритетов. Их можно вызвать из программы.

Коды opalu показаны в таблице 6:

Таблица 6

opalu	Выполняемая функция	Примечание
x0	OR (RD = RA OR RB / Imm)	ИЛИ
x1	AND (RD = RA AND RB / Imm)	И
x2	XOR (RD = RA XOR RB / Imm)	Исключающее ИЛИ
x3	ANDN (RD = RA ANDN RB / Imm)	Сброс бита
x4	NOR (RD = NOT(RA OR RB / Imm)	ИЛИ НЕ
x5	BIT RA (RD = RA BIT RB / Imm)	Проверка бита
x6	SET RA (RD = RA SET RB / Imm)	Установка бита
x7	RES RA (RD = RA RES RB / Imm)	Сброс бита
x8	ADD (RD = RA + RB / Imm)	Сложение без переноса
x9	ADC (RD = RA + RB / Imm + C)	Сложение с переносом
xA	SUB (RD = RA – RB / Imm)	Вычитание без заема
xB	SBC (RD = RA – RB / Imm - C)	Вычитание с заемом
xC	SRA (RD = RA >> RB / Imm)	Сдвиг вправо арифметич.
xD	SRL (RD = RA >> RB / Imm)	Сдвиг вправо логический
xE	SLL (RD = RA << RB / Imm)	Сдвиг влево логический
xF	ABS RA (RD = ABS RA)	Абсолютное значение
x10	ROL	Цикл.сдвиг на 1 р.
x11	ROR	Цикл.сдвиг на 1 р.
x12	RCL	Цикл.сдвиг на 1 р. через С
x13	RCR	Цикл.сдвиг на 1 р. через С
x14		
x15		
x16		
x17		
x18		
x19		
x1A		
x1B		
x1C		
x1D		
x1E	SMUL (RD = RA * RB / Imm)	Умножение знаковое
x1F	SDIV	Деление знаковое
x20..x3F		Резерв

Красным цветом выделены команды, которые не обязательно реализовывать или можно заменить на другие.