Datasheet do ZR16 Softcore

1. Visão geral

O ZR16 softcore é um microprocessador descrito em VHDL de 8 bits com memória de programa de 1024 palavras, e memória RAM de 256 bytes.

2. Características

- 24 instruções;
- 16 registradores,
- Pilha de 4 posições de contexto para armazenamento.
- 8 bits de largura do barramento de dados.
- Memória de programa FLASH de 1024 x 16 bits.
- Memória de dados de 256 x 8 bits.

3. Arquitetura

O ZR16 softcore é um microprocessador de bits, RISC usando arquitetura Harvard, com 24 instruções. A mem´oria de programa é do tipo FLASH externa. O microprocessador possui banco de 16 registradores sendo 13 de uso geral. Os outros 3 registradores contêm o program counter (PC, endereço atual de execução do programa), os controles e os flags. Possui um stack de 4 posições para armazenar flags, controles e PC.

4. Espaços de Endereçamento

4.1. Banco de registradores

O ZR16 softcore contém 16 registradores, entre os quais 13 são de uso geral. Estes registradores podem ser endereçados nas instruções diretamente por modos específicos de endereçamento.

Registrador r15:

- Z: flag do sistema: Z=1 se o resultado da operação for zero.
- C: flag do sistema: informa se houve um Carry ou Borrow nas operações aritméticas (instruções ADD, SUB e COMP). Para as instruções ROT, SHL e SHA é o bit final resultado do deslocamento.
- VP: flag do sistema: VP= 1 quando o resultado de uma operação aritmética (ADD, SUB OU CMP) não puder ser representado em complemento de 2, ou quando o número de bits em 1 do resultado de uma operação lógica (AND, OR ou XOR) for par.
- uf1 a uf0: flags do usuário.
- uc: controle: quando uc = 1 o carry é usado nas instruções ADD, SUB, CMP e ROT.
- e/d: controle: usado nas instruções de ROT, SHL e SHA para informar se o deslocamento dos bits é para esquerda (e/d=1) ou direita (e/d=0).
- hint: controle: quando hint = 1 interrupções

Registrador r14:

- mp: informa, quando estão sendo usado as instruções MOV rd, (ro) ou MOV r0, (end), se os dados são oriundos da memória de dados (mp=0) ou da memória de programa (mp=1).
- p2 a p0: quando mp=1, aponta a partição da memória de programa a ser acessada. São 8 partições de 128 x 16 bits.
- ppc9 e ppc8: são os próximos valores dos bits pc8 e pc9, se for usada uma instrução que altera o Program Counter (quando rd = r13), carregando diretamente o valor de um registrador, ou usando o dado apontado por um registrador.

Registrador r13:

- pc7 a pc0: Bits 7 a 0 do Program Counter.

4.2. Pilha de contexto do programa

Além dos 16 registradores o ZR16S08 possui 4 posições de pilha de contexto de programa (stack), onde são armazenadas as informações de ambiente no instante de execução de uma exceção.

São consideradas exceções o salto para uma sub-rotina, usando a instrução CALL. O retorno para o fluxo de execução principal se dá pela instrução RET e suas variações. São armazenados na pilha:

- registrador r13
- registrador r14
- registrador r15 - com exceção do bit hint

Quando ocorre uma exceção há um empilhamento, ou seja, o conteúdo dos registradores r15, r14 e r13 são armazenados no Contexto(0). O conteúdo de cada Contexto é salvo no Contexto de nível imediatamente superior. O conteúdo do Contexto(3) é perdido, por esta razão só é possível ter o contexto de 4 exceções empilhadas, sem que se execute um desempilhamento.

O desempilhamento dos contextos se dá na execução da instrução RET e de suas variações. Quando uma instrução RET é executada, o conteúdo é passado para os registradores r15, r14 e r13. Cada Contexto salva o conteúdo do Contexto imediatamente superior. O Contexto 3 tem o seu conteúdo zerado. Assim, existindo mais retornos de contexto (execução de RET) do que exceções o programa retornará à sua posição inicial (0x000).

A conteúdo da pilha só pode ser recuperado através da instrução RET e de suas variações. Só está disponível o contexto atual, ou seja, o conteúdo dos registradores r15, r14 e r13.

4.3. Memória de programa

O microcontrolador contém uma memória de programa externa, não volátil e programável, uma FLASH.

A capacidade da memória de programa do ZR16 softcore é de 1024 palavras de 16 bits. O primeiro endereço é a posição 0x000 que é o ponto no qual o Program Counter inicia logo após o reset do microprocessador.

É possível ler a memória de programa a partir de instruções do ZR16 softcore. Para fins de acesso como dados a memória de programa é particionada em 8 partes cada uma de 128 palavras de 16 bits. As instruções que acessam a memória do programa como dados são: mov r0,(end) e mov rd,(ro), sendo que para isto o bit mp de r14 tem que ser igual 1.

4.4. Memória de dados

A memória de dados é uma RAM com capacidade de 256 x 8 bits (256 Bytes), é volátil, os dados são perdidos após um reset ou do desligamento do microprocessador.

5. Instruções e Endereçamentos

As instruções do ZR16 softcore são apresentadas na Tabela 1.

Tabela 1 - Instruções implementadas.

Instrução	Opcode
JMP	0000
JZ, JNZ, JC, JVP	0001
CALL	0010
RET, RETC, RETS, RETZ	0011 e bit 7 =1
MVS	0011 e bit 7 =0
AND	0100
OR	0101
XOR	0110
CMP	0111
ADD	1000
SUB	1001
ROT	1010
SHL	1011
SHA	1100
MOV	1101
DJNZ	1110
INC	1111 e bit 8 =0
DEC	1111 e bit 8 =1

A tabela 2 lista todas as instruções implementadas em todos os modos de endereçamento permitidos indicando os flags afetados e o número de ciclos de duração de cada instrução.

Tabela 2 - Instruções implementadas com os modos de endereçamento.

			opcode																	
mp	**ro0/end0**	**Instrução**	**b15**	**b14**	**b13**	**b12**	**b11**	**b10**	**b9**	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**	**flags**	**cycles**
0	-	MOV rd, ro	1	1	0	1	0	0	0	0	rd				ro				Z	1
0	-	MOV rd, (ro)	1	1	0	1	0	0	0	1	rd				ro				Z	2
1	0	MOV rd, (ro)	1	1	0	1	0	0	0	1	rd				ro				Z	3
1	1	MOV rd, (ro)	1	1	0	1	0	0	0	1	rd				ro				Z	3
0	-	MOV r0, (end)	1	1	0	1	0	0	1	0	endereço								Z	2
1	0	MOV r0, (end)	1	1	0	1	0	0	1	0	endereço								Z	3
1	1	MOV r0, (end)	1	1	0	1	0	0	1	0	endereço								Z	3
0	-	MOV r0, imediato	1	1	0	1	0	0	1	1	imediato								Z	1
0	-	MOV (rd), ro	1	1	0	1	0	1	0	0	rd				ro				Z	1
0	-	MOV (rd),(ro)	1	1	0	1	0	1	0	1	rd				ro				Z	2
0	-	MOV (r0), (end)	1	1	0	1	0	1	1	0	endereço								Z	2
0	-	MOV (r0), imediato	1	1	0	1	0	1	1	1	imediato								Z	1
0	-	MOV (end), r0	1	1	0	1	1	0	0	0	endereço								Z	1
0	-	MOV (end), (r0)	1	1	0	1	1	0	0	1	endereço								Z	2
0	-	MVS rd, imediato	0	0	1	1	rd				0	imediato							Z	1
0	-	ADD rd, ro	1	0	0	0	0	0	0	0	rd				ro				Z	1
0	-	ADD rd, (ro)	1	0	0	0	0	0	0	1	rd				ro				Z	2
0	-	ADD r0, (end)	1	0	0	0	0	0	1	0	endereço								Z	2
0	-	ADD r0, imediato	1	0	0	0	0	0	1	1	imediato								Z	1
0	-	ADD (rd), ro	1	0	0	0	0	1	0	0	rd				ro				Z	1
0	-	ADD (rd),(ro)	1	0	0	0	0	1	0	1	rd				ro				Z	2
0	-	ADD (r0), (end)	1	0	0	0	0	1	1	0	endereço								Z	2
0	-	ADD (r0), imediato	1	0	0	0	0	1	1	1	imediato								Z	1
0	-	ADD (end), r0	1	0	0	0	1	0	0	0	endereço								Z	1
0	-	ADD (end), (r0)	1	0	0	0	1	0	0	1	endereço								Z	2
0	-	SUB rd, ro	1	0	0	1	0	0	0	0	rd				ro				Z	1
0	-	SUB rd, (ro)	1	0	0	1	0	0	0	1	rd				ro				Z	2
0	-	SUB r0, (end)	1	0	0	1	0	0	1	0	endereço								Z	2
0	-	SUB r0, imediato	1	0	0	1	0	0	1	1	imediato								Z	1
0	-	SUB (rd), ro	1	0	0	1	0	1	0	0	rd				ro				Z	1
0	-	SUB (rd),(ro)	1	0	0	1	0	1	0	1	rd				ro				Z	2
0	-	SUB (r0), (end)	1	0	0	1	0	1	1	0	endereço								Z	2
0	-	SUB (r0), imediato	1	0	0	1	0	1	1	1	imediato								Z	1
0	-	SUB (end), r0	1	0	0	1	1	0	0	0	endereço								Z	1

0	-	SUB (end), (r0)	1	0	0	1	1	0	0	1	endereço		Z	2

0	-	CMP rd, ro	0	1	1	1	0	0	0	0	rd	ro	Z	1
0	-	CMP rd, (ro)	0	1	1	1	0	0	0	1	rd	ro	Z	2
0	-	CMP r0, (end)	0	1	1	1	0	0	1	0	endereço		Z	2
0	-	CMP r0, imediato	0	1	1	1	0	0	1	1	imediato		Z	1
0	-	CMP (rd), ro	0	1	1	1	0	1	0	0	rd	ro	Z	1
0	-	CMP (rd),(ro)	0	1	1	1	0	1	0	1	rd	ro	Z	2
0	-	CMP (r0), (end)	0	1	1	1	0	1	1	0	endereço		Z	2
0	-	CMP (r0), imediato	0	1	1	1	0	1	1	1	imediato		Z	1
0	-	CMP (end), r0	0	1	1	1	1	0	0	0	endereço		Z	1
0	-	CMP (end), (r0)	0	1	1	1	1	0	0	1	endereço		Z	2

0	-	AND rd, ro	0	1	0	0	0	0	0	0	rd	ro	Z	1
0	-	AND rd, (ro)	0	1	0	0	0	0	0	1	rd	ro	Z	2
0	-	AND r0, (end)	0	1	0	0	0	0	1	0	endereço		Z	2
0	-	AND r0, imediato	0	1	0	0	0	0	1	1	imediato		Z	1
0	-	AND (rd), ro	0	1	0	0	0	1	0	0	rd	ro	Z	1
0	-	AND (rd),(ro)	0	1	0	0	0	1	0	1	rd	ro	Z	2
0	-	AND (r0), (end)	0	1	0	0	0	1	1	0	endereço		Z	2
0	-	AND (r0), imediato	0	1	0	0	0	1	1	1	imediato		Z	1
0	-	AND (end), r0	0	1	0	0	1	0	0	0	endereço		Z	1
0	-	AND (end), (r0)	0	1	0	0	1	0	0	1	endereço		Z	2

0	-	OR rd, ro	0	1	0	1	0	0	0	0	rd	ro	Z	1
0	-	OR rd, (ro)	0	1	0	1	0	0	0	1	rd	ro	Z	2
0	-	OR r0, (end)	0	1	0	1	0	0	1	0	endereço		Z	2
0	-	OR r0, imediato	0	1	0	1	0	0	1	1	imediato		Z	1
0	-	OR (rd), ro	0	1	0	1	0	1	0	0	rd	ro	Z	1
0	-	OR (rd),(ro)	0	1	0	1	0	1	0	1	rd	ro	Z	2
0	-	OR (r0), (end)	0	1	0	1	0	1	1	0	endereço		Z	2
0	-	OR (r0), imediato	0	1	0	1	0	1	1	1	imediato		Z	1
0	-	OR (end), r0	0	1	0	1	1	0	0	0	endereço		Z	1
0	-	OR (end), (r0)	0	1	0	1	1	0	0	1	endereço		Z	2

0	-	XOR rd, ro	0	1	1	0	0	0	0	0	rd	ro	Z	1
0	-	XOR rd, (ro)	0	1	1	0	0	0	0	1	rd	ro	Z	2
0	-	XOR r0, (end)	0	1	1	0	0	0	1	0	endereço		Z	2
0	-	XOR r0, imediato	0	1	1	0	0	0	1	1	imediato		Z	1
0	-	XOR (rd), ro	0	1	1	0	0	1	0	0	rd	ro	Z	1
0	-	XOR (rd),(ro)	0	1	1	0	0	1	0	1	rd	ro	Z	2
0	-	XOR (r0), (end)	0	1	1	0	0	1	1	0	endereço		Z	2
0	-	XOR (r0), imediato	0	1	1	0	0	1	1	1	imediato		Z	1
0	-	XOR (end), r0	0	1	1	0	1	0	0	0	endereço		Z	1
0	-	XOR (end), (r0)	0	1	1	0	1	0	0	1	endereço		Z	2

0	-	INC rd	1	1	1	1	0	0	-	0	rd		Z	1
0	-	INC (rd)	1	1	1	1	0	1	-	0	rd		Z	1
0	-	INC (end)	1	1	1	1	1	0	-	0	endereço		Z	1

0	-	DEC rd	1	1	1	1	0	0	-	1	rd		Z	1
0	-	DEC (rd)	1	1	1	1	0	1	-	1	rd		Z	1
0	-	DEC (end)	1	1	1	1	1	0	-	1	endereço		Z	1

0	-	ROT rd, ro	1	0	1	0	0	0	0	0	rd	ro	Z	1
0	-	ROT rd, (ro)	1	0	1	0	0	0	0	1	rd	ro	Z	2
0	-	ROT r0, (end)	1	0	1	0	0	0	1	0	endereço		Z	2
0	-	ROT (rd), ro	1	0	1	0	0	1	0	0	rd	ro	Z	1
0	-	ROT (rd),(ro)	1	0	1	0	0	1	0	1	rd	ro	Z	2
0	-	ROT (r0), (end)	1	0	1	0	0	1	1	0	endereço		Z	2
0	-	ROT (end), r0	1	0	1	0	1	0	0	0	endereço		Z	1
0	-	ROT (end), (r0)	1	0	1	0	1	0	0	1	endereço		Z	2

0	-	SHL rd, ro	1	0	1	1	0	0	0	0	rd	ro	Z	1
0	-	SHL rd, (ro)	1	0	1	1	0	0	0	1	rd	ro	Z	2
0	-	SHL r0, (end)	1	0	1	1	0	0	1	0	endereço		Z	2
0	-	SHL (rd), ro	1	0	1	1	0	1	0	0	rd	ro	Z	1
0	-	SHL (rd),(ro)	1	0	1	1	0	1	0	1	rd	ro	Z	2
0	-	SHL (r0), (end)	1	0	1	1	0	1	1	0	endereço		Z	2
0	-	SHL (end), r0	1	0	1	1	1	0	0	0	endereço		Z	1
0	-	SHL (end), (r0)	1	0	1	1	1	0	0	1	endereço		Z	2

0	-	SHA rd, ro	1	1	0	0	0	0	0	0	rd	ro	Z	1
0	-	SHA rd, (ro)	1	1	0	0	0	0	0	1	rd	ro	Z	2
0	-	SHA r0, (end)	1	1	0	0	0	0	1	0	endereço		Z	2
0	-	SHA (rd), ro	1	1	0	0	0	1	0	0	rd	ro	Z	1
0	-	SHA (rd),(ro)	1	1	0	0	0	1	0	1	rd	ro	Z	2
0	-	SHA (r0), (end)	1	1	0	0	0	1	1	0	endereço		Z	2
0	-	SHA (end), r0	1	1	0	0	1	0	0	0	endereço		Z	1
0	-	SHA (end), (r0)	1	1	0	0	1	0	0	1	endereço		Z	2

0	-	JMP par rd	0	0	0	0	0	0	par		rd	----	Z	2
0	-	JMP par (rd)	0	0	0	0	0	1	par		rd	----	Z	2
0	-	JMP end10	0	0	0	0	1	-	end10				Z	2

0	-	JZ end10	0	0	0	1	0	0	end10				Z	2
0	-	JNZ end10	0	0	0	1	0	1	end10				Z	2
0	-	JC end10	0	0	0	1	1	0	end10				Z	2
0	-	JVP end10	0	0	0	1	1	1	end10				Z	2

0	-	CALL par rd	0	0	1	0	0	0	par	rd		----						Z	2
0	-	CALL par (rd)	0	0	1	0	0	1	par	rd		----						Z	2
0	-	CALL end10	0	0	1	0	1	-	end10									Z	2

0	-	RET	0	0	1	1					1						0	0	Z	1
0	-	RETC	0	0	1	1					1						0	1	Z	1
0	-	RETZ	0	0	1	1					1						1	0	Z	1
0	-	RETS	0	0	1	1					1						1	1	Z	1

0	-	DJNZ	1	1	1	0	rd	end10	Z	3

O ZR16S08 possibilita realizar operações com os registradores de controle R13, R14 e R15, e quando o destino dessas operações for os registradores e controle, elas terão um comportamento específico.

Caso destino da operação lógica seja r15 os flags Z, C e V P serão carregadas com o valor do registrador transferido para estes bits.

MOV

Na instrução MOV, o conteúdo é copiado do operando de origem para o destino. O valor da origem é mantido e o valor do destino é sobrescrito. Quando r14!=1 (mp) e for executada a instrução MOV r0, (end) ou MOV rd, (ro) é transferido um dado da memória de programa para o registrador especificado. O flag Z é acertado de acordo com o resultado da movimentação. Os flags C e V P não são afetados.

MVS

A instrução MVS carrega em registrador (r0 a r15) com o valor imediato contido nos bits 6 a 0 da instrução (valor entre: 0 a 7F). O flag Z é carregado de acordo com o resultado da transferência. Os flags C e V P não são afetados. Caso o destino da operação lógica seja r15, neste caso Z será sempre carregado com '0'.

ADD, SUB

A instrução [ADD, SUB] efetua a adição (ADD) ou subtração (SUB) de dois operandos. O resultado é carregado no operando destino.

CMP

A instrução CMP executa a operação de subtração do operando destino pelo operando origem. Na instrução CMP o destino fica inalterado, só são acertados os flags Z, C e V P de acordo com o resultado da operação.

INC, DEC

A instrução [INC, DEC] incrementa (INC) ou decrementa (DEC) o valor do operando que pode ser um registrador ou valor do dado da memória de dados. O flag Z é acertado de acordo com o resultado da movimentação. Os flags C e V P não são afetados.

AND, OR, XOR

A instrução [AND, OR, XOR] efetua a operação lógica [AND, OR, XOR], bit a bit com o operando destino e o operando origem. O resultado é carregado no operando destino. O flag Z é acertado de acordo com o resultado da operação. O flag C não é afetado pela operação. O flag V P é carregado de acordo com a paridade do resultado, '1' se o número de bits em '1' do resultado for par, senão '0'.

ROT

A instrução ROT faz a operação de rotação lógica de uma posição do dado de origem, pondo o resultado da rotação no destino. A rotação para esquerda ou para direita é definido por $r15_1$(e/d). Se $r15_1$= 1 roda para esquerda, senão direita. O uso do flag C na rotação é definido por $r15_2$(uc). Se $r15_2$=1 usa C senão não usa. Os flags Z e C são acertados de acordo com o resultado da operação. O flag V P não é afetado.

SHL, SHA

A instrução [SHL, SHA], executa o deslocamento lógico [SHL] ou aritmético [SHA] de um bit para a esquerda ou para a direita, do dado origem, pondo o resultado no destino. O deslocamento para esquerda ou para direita é definido por $r15_1$(e/d). Se $r15_1$= 1 desloca para esquerda, senão direita. Os flags Z e C são acertados de acordo com o resultado da operação. Quando a instrução for SHA o bit V P também tem seu valor definido de acordo com o resultado da operação.

JMP

A instrução JMP permite ao microprocessador saltar a execução do programa para um endereço determinado. O endereço de destino pode ser o conteúdo de um registrador, o conteúdo do endereço da memória de dados apontado por um registrador ou um endereço absoluto fornecido na própria palavra da instrução.

No caso de o endereço destino ser o conteúdo do endereço da memória de dados apontado por um registrador é feito um acesso de leitura na memória de dados para obter o conteúdo do endereço apontado pelo registrador.

Quando o endereço ´e fornecido pelo conteúdo de um registrador ou pelo conteúdo do endereço de memória de dados apontado por um registrador, a memória de programa é considerada como se tivesse 4 partições de 256 endereços. Os bits que informam a partição (p1 e p0) são fornecidos na instrução e são usados como bits 9 e 8 no PC.

JCC: JZ, JNZ, JC, JVP

Quando da execução da instrução Jcc o microprocessador salta para um endereço determinado se o teste do flag especificado for verdadeiro. O endereço de salto ´e fornecido na própria instrução.

- JZ salta quando Z = 1
- JNZ salta quando Z = 0
- JC salta quando C = 1
- JVP salta quando V_P = 1

CALL

A instrução CALL possibilita ao microprocessador saltar a execução do programa para um endereço determinado, salvando na pilha de contexto o ambiente de execução em que estava (PC, controles e flags), conforme a figura 3. O endereço de destino pode ser o conteúdo de um registrador, o conteúdo do endereço da memória de dados apontado por um registrador ou um endereço absoluto fornecido na própria palavra da instrução.

Quando o endereço destino é o conteúdo do endereço da memória de dados apontado por um registrador, ´e feito um acesso de leitura na memória de dados para obter o conteúdo do endereço apontado pelo registrador. Quando o endereço é fornecido pelo conteúdo de um registrador ou pelo conteúdo do endereço de memória de dados apontado por um registrador,
a memória de programa ´e considerada como se tivesse 4 partições de 256 endereços. Os bits que informam a partição (p1 e p0) são fornecidos na instrução e são usados como bits 9 e 8 no PC.
Os flags Z, C e V P não são afetados quando da execução desta instrução.

RET

A instrução RET é usada para retornar o ambiente original de execução (PC, controles e flags). Ela é usada tanto no caso de interrupção, como para retornar de chamada de sub-rotina (CALL). A instrução RET possibilita a programação do bit hint ($r15_0$). $r15_0$ pode ser mantido como

está (RET), zerado (RETZ), ligado (RETS) ou complementado (RETC). Durante a execução da instrução RET, mesmo que haja uma interrupção e hint = 1 (habilitação de interrupção), não ocorre o aceite de interrupção.

DJNZ

A instrução DJNZ decrementa um registrador especificado (r1, r2, r3 ou r4) e verifica se o resultado é zero. Se o resultado for zero, executa a próxima instrução, senão salta para o endereço especificado na instrução. O flag Z acertado de acordo com o resultado do decremento do registrador. Os flags C e V P permanecem inalterados.