MaxCPU (TU)

1. External Interface

1.1. Generic Parameters

1.1.1. The data width is generic with a minimum of 12 bit.

1.1.2. The address width is generic with a minimum of 12 bit which implies a total number of 2^(address width) addressable memory cells.

1.2. Ports

1.2.1. A clock input (processor is triggered on the positive clock edge).

1.2.2. A reset input (asynchronous, active low).

1.2.3. A data bus from memory or I/O-devices to processor, size according to 1.1.1.

1.2.4. A data bus from processor to memory, size according to 1.1.1.

1.2.5. An address bus from processor to memory, size according to 1.1.2.

1.2.6. A 1-Bit control signal to switch between data from memory (if ‘0’) and I/O-devices (if ‘1’).

1.2.7. A 1-Bit control signal to indicate write-access to the memory (active high).

1.2.8. A 1-Bit control signal to indicate access to the I/O-devices (active high).

1.2.9. A 1-Bit control signal to distinguish between read- (if ‘0’) and write-access (if ‘1’) to the I/O-devices.

1.2.10. A 1-Bit control signal to indicate if the addressed I/O-device is ready to send/receive data to/from the processor (active high).

2. Structural Description

2.1. Internal Registers

2.1.1. Code and Data are accessed in the same address space (“Von Neumann” Architecture).

2.1.2. Four data registers

2.1.2.1. The size of these registers is based on the general data width (according to 1.1.1).

2.1.2.2. The initial value of these registers is 0x0.

2.1.2.3. The registers contain operands and results from operations; there is no restriction regarding the use of registers.

2.1.2.4. The content of all registers not currently used as destination of an instruction won’t change their value.

2.1.3. Program Counter (PC)

2.1.3.1. The size of the PC is based on the general address width (according to 1.1.2).

2.1.3.2. The initial value of the PC is 0x0.

2.1.3.3. After an instruction has been fetched from the code memory, PC is incremented by one and thus contains the address of the next instruction to be fetched.

2.1.3.4. After the PC reaches its maximum value (2^(address width) – 1), the next value is 0x0.

2.2. System Reset

2.2.1. A low level on the reset input of the processor causes the system to go in a defined state (asynchronous reset, active low).

2.2.2. Data registers and PC return to their initial states (according to points 2.1.2.2 and 2.1.3.2).

2.2.3. The internal flags return to their initial state (according to points 3.2.4.2, 3.2.5.7, 3.2.6.3 and 3.2.7.4).

3. Instructions

3.1. General Issues

3.1.1. All bits of an instruction following the OpCode (size 6 bit) and the necessary parameters to the end of the instruction are ‘don’t care’.

3.1.2. The instructions are distributed as follows (see point 3.3 for details):

3.1.2.1. Miscellaneous Instructions (2): NOP, STOP.

3.1.2.2. Arithmetic Instructions (4): ADD, ADDC, SUB, SUBC.

3.1.2.3. Logical Instructions (7): NOT, AND, OR, XOR, REA, REO, REX.

3.1.2.4. Shift / Rotate Instructions (7): SLL, SRL, SRA, ROL, ROLC, ROR, RORC.

3.1.2.5. Memory Access Instructions (5): LDC, LDD, LDR, STD, STR.

3.1.2.6. I/O Instructions (2): IN, OUT.

3.1.2.7. Jump Instructions (9): JMP, JZ, JC, JN, JO, JNZ, JNC, JNN, JNO.

3.2. Internal Flags

3.2.1. In order to store particular results of some operations, there are 4 internal Flags: Zero, Carry, Negative and Overflow.

3.2.2. These flags are bits that can contain the values ‘0’ (the flag is cleared) or ‘1’ (the flag is set).

3.2.3. The flags are only affected when a data word is written into a register (that means by the following instructions: ADD, ADDC, SUB, SUBC, NOT, AND, OR, XOR, REA, REO, REX, SLL, SRL, SRA, ROL, ROLC, ROR, RORC, LDC, LDD, LDR, IN).

3.2.4. Zero Flag

3.2.4.1. When executing one of the instructions listed in 3.2.3, the result is evaluated; if the value equals 0x0, the zero flag is set, otherwise it is cleared.

3.2.4.2. The initial value of the zero flag is ‘0’.

3.2.5. Carry Flag

3.2.5.1. The carry flag is cleared by the following instructions: NOT, AND, OR, XOR, REA, REO, REX, ROL, ROR.

3.2.5.2. When executing the ADD or ADDC instruction, the carry output of an unsigned addition of the two source operands is assigned to the carry flag.

3.2.5.3. When executing the SUB or SUBC instruction, the carry output of an unsigned addition of the first source operand and the two’s complement of the second source operand is assigned to the carry flag.

3.2.5.4. When executing the SLL or ROLC instruction, the most significant bit of the source operand is assigned to the carry flag.

3.2.5.5. When executing the SRL, SRA or RORC instruction, the least significant bit of the source operand is assigned to the carry flag.

3.2.5.6. LDC, LDD, LDR and IN don’t affect the carry flag.

3.2.5.7. The initial value of the carry flag is ‘0’.

3.2.6. Negative Flag

3.2.6.1. When executing the ADD, ADDC, SUB or SUBC instruction, the source operands are sign-extended by one bit; the additional bit of the result is assigned to the negative flag.

3.2.6.2. When executing one of the remaining instructions listed in 3.2.3, the most significant bit of the result is assigned to the negative flag.

3.2.6.3. The initial value of the negative flag is ‘0’.

3.2.7. Overflow Flag

3.2.7.1. When executing the ADD, ADDC, SUB or SUBC instruction, the source operands are sign-extended by one bit; if the additional bit of the result differs from the (original) most significant bit of the result, the overflow flag is set, otherwise it is cleared.

3.2.7.2. When executing the SLL instruction, the overflow flag is set if the most significant bit of the source operand differs from the most significant bit of the result, otherwise it is cleared.

3.2.7.3. When executing one of the remaining instructions listed in 3.2.3, the overflow flag is cleared.

3.2.7.4. The initial value of the overflow flag is ‘0’.

3.3. Detailed Instruction Set

3.3.1. The following table presents the detailed instruction set of the processor, the syntax and the function that must be realized for each instruction:

Instruction Class	Spec. No.	Syntax		Parameter	Instruction Format	Action	Instr.Code
Miscellaneous	3.3.1.1	NOP		n/a	Format 1	No operation	0
	3.3.1.2	STOP		n/a	Format 1	Stop processor	1
Arithmetic	3.3.1.3	ADD	D S1 S2	n/a	Format 1	D := S1 + S2	2
	3.3.1.4	ADDC	D S1 S2	n/a	Format 1	D := S1 + S2 + Carry	3
	3.3.1.5	SUB	D S1 S2	n/a	Format 1	D := S1 – S2	4
	3.3.1.6	SUBC	D S1 S2	n/a	Format 1	D := S1 – S2 – Carry	5
Logical	3.3.1.7	NOT	D S1 S2	n/a	Format 1	D := NOT S1	6
	3.3.1.8	AND	D S1 S2	n/a	Format 1	D := S1 AND S2	7
	3.3.1.9	OR	D S1 S2	n/a	Format 1	D := S1 OR S2	8
	3.3.1.10	XOR	D S1 S2	n/a	Format 1	D := S1 XOR S2	9
	3.3.1.11	REA	D S1 S2	n/a	Format 1	LSB(D) := reduced_and(S1)	10
	3.3.1.12	REO	D S1 S2	n/a	Format 1	LSB(D) := reduced_or(S1)	11
	3.3.1.13	REX	D S1 S2	n/a	Format 1	LSB(D) := reduced_xor(S1)	12
Shift / Rotate	3.3.1.14	SLL	D S1 S2	n/a	Format 1	Carry & D := S1 & ‘0’	13
	3.3.1.15	SRL	D S1 S2	n/a	Format 1	D & Carry := ‘0’ & S1	14
	3.3.1.16	SRA	D S1 S2	n/a	Format 1	D & Carry := MSB(S1) & S1	15
	3.3.1.17	ROL	D S1 S2	n/a	Format 1	D := rotate_left(S1)	16
	3.3.1.18	ROLC	D S1 S2	n/a	Format 1	Carry & D := rotate_left(Carry & S1)	17
	3.3.1.19	ROR	D S1 S2	n/a	Format 1	D := rotate_right(S1)	18
	3.3.1.20	RORC	D S1 S2	n/a	Format 1	Carry & D := rotate_right(Carry & S1)	19
Memory Access	3.3.1.21	LDC	D	const	Format 2	D := const	32
	3.3.1.22	LDD	D	addr	Format 2	D := mem(addr)	33
	3.3.1.23	LDR	D RA	n/a	Format 2	D := mem(RA)	34
	3.3.1.24	STD	S	addr	Format 2	mem(addr) := S	35
	3.3.1.25	STR	S RA	n/a	Format 2	mem(RA) := S	36
I/O	3.3.1.26	IN	D	n/a	Format 2	D := data_from_input_device	37
	3.3.1.27	OUT	S	n/a	Format 2	data_to_output_device := S	38
Jump	3.3.1.28	JMP		addr	Format 1	Unconditional Jump	48
	3.3.1.29	JZ		addr	Format 1	Jump if zero flag = ‘1’	49
	3.3.1.30	JC		addr	Format 1	Jump if carry flag = ‘1’	50
	3.3.1.31	JN		addr	Format 1	Jump if negative flag = ‘1’	51
	3.3.1.32	JO		addr	Format 1	Jump if overflow flag = ‘1’	52
	3.3.1.33	JNZ		addr	Format 1	Jump if zero flag = ‘0’	53
	3.3.1.34	JNC		addr	Format 1	Jump if carry flag = ‘0’	54
	3.3.1.35	JNN		addr	Format 1	Jump if negative flag = ‘0’	55
	3.3.1.36	JNO		addr	Format 1	Jump if overflow flag = ‘0’	56

Note: If an instruction needs a parameter it is located in the memory cell directly following that holding the instruction.

3.3.2. The instruction word is composed of the 6 bit OpCode and following 6 bits reserved for destination and source register specification (as shown in the table above; if only one register has to be specified, it has to be in the first position). In case of a data width larger than 12 the bits following the 12 bit instruction word are ‘don’t care’.

3.3.3. The processor is able to work with unsigned and signed numbers. The carry flag is always computed interpreting the operands as unsigned, while the negative and overflow flags are always computed interpreting the operands as signed.

3.3.4. For REA, REO and REX operations the result is stored in the least significant bit while all other bits of the destination register are cleared.

3.3.5. When an unknown OpCode is decoded, it is treated as a NOP instruction and the user is informed of the illegal OpCode by an assertion.

3.3.6. When a STOP instruction is decoded the processor stops fetching new instructions and reaches an explicit deadlock state. In this state there are no changes to any registers (data registers, flags, PC) and no communication to memory or I/O devices.

3.4. I/O-Instructions

The IN instruction causes the input-device to send the content of its data register to the processor if the device is ready; if the device is not ready, the processor is blocked until the input-device is ready again.

The OUT instruction causes the output-device to receive a data word from the processor and store it in its data register if the device is ready; if the device is not ready, the processor is blocked until the output-device is ready again.

3.5. I/O-Devices

3.5.1. General Structure

All I/O-devices contain two registers, one register for storing a data word (size according to 1.1.1), and a one-bit registers storing the information if the data register is empty or not.

3.5.2. The external interface of the I/O-devices is composed of:

3.5.2.1. A Clock input (I/O-devices are triggered on the positive clock edge).
3.5.2.2. A Reset input (asynchronous, active low).
3.5.2.3. A control signal to indicate if the I/O-device is ready to send/receive data to/from the processor (active high).
3.5.2.4. A control signal to indicate access to the I/O-device (active high).
3.5.2.5. A control signal to distinguish between read- (if ‘0’) and write-access (if ‘1’) to the I/O-device.
3.5.2.6. A data bus from the processor (only output device, size according to 1.1.1).
3.5.2.7. A data bus to the outside (only output device; size according to 1.1.1).
3.5.2.8. A data ready signal to the outside (only output device).
3.5.2.9. A data request signal from the outside (only output device).
3.5.2.10. A data bus to the processor (only input device, size according to 1.1.1).
3.5.2.11. A data bus from the outside (only input device; size according to 1.1.1).
3.5.2.12. A data ready signal from the outside (only input device).
3.5.2.13. A data request signal to the outside (only input device).

3.5.3. Input device

The device includes a register for storing a data word from the outside bus to be sent to the processor.

If the internal register contains data, the DevRdy signal of this device has to be set (since the device is now ready to send this data to the processor); otherwise it has to be cleared.

If the AccEn input is set, the AccType input is set to ‘0’ and the internal register contains data, this data has to be made visible at the port DataOut; in all other cases DataOut has to be set to 0x0.

If the internal register is empty, the DataReq signal of this device has to be set; otherwise it has to be cleared.

If the DataRdy input is set and the internal register is empty, the data at the port DataIn has to be stored in the internal register; in all other cases the content of the internal register must not change.

3.5.4. Output device

The device includes a register for storing a data word from the processor to be sent to the outside bus.

If the internal register is empty, the DevRdy signal of this device has to be set (since the device is now ready to receive data from the processor); otherwise it has to be cleared.

If the AccEn input is set, the AccType input is set to ‘1’ and the internal register is empty, the data at the port DataIn has to be stored in the internal register; in all other cases the content of the internal register must not change.

If the DataReq input is set and the internal register contains data, this data has to be made visible at the port DataOut and DataRdy has to be set; in all other cases DataOut has to be set to 0x0 and DataRdy has to be cleared.