

LU computer architecture

assignment

Instruction Set II

Philipp Effenberger, Matr. Nr.0326017, E938

e0326017 <at> student tuwien ac at

Markus Kammerstetter, Matr. Nr.0226196, E535

e0226196 <at> student tuwien ac at

Armin Novak, Matr. Nr.e0225426, E938

e0225426 <at> student tuwien ac at

Gerald Scharitzer, Matr. Nr.e0127228

e0127228 <at> student tuwien ac at

Vienna, Tuesday 6th November 2007

1 assignment

Define an instruction set for your processor. Define the encoding of the instructions and develop a simple assembler.

Hint: You don't have to go fancy with macros or several file linking with your assembler. Use a simple single file assembler and let the C preprocessor do the work. The command line options for the GNU C compiler are:

```
gcc -x c -E -C -P infile.asm > outfile.asm
```

2 **Instruction set for MAD**

We decided to build an 16 bit RISC based load / store architecture.

2.1 **Basic Layout**

- 16 bit RISC CPU
- Von Neumann architecture
- 16 Bit bus
- 15 General Purpose Registers + Stack Pointer
- big endian byte order
- ALU operations
- 4 stage pipeline

2.2 **Registers**

All registers are 16 bits long. Only the general purpose registers and the stack pointer can be specified as register operands. The other registers are implicitly accessed by designated instructions.

2.2.1 **General Purpose Registers**

The processor provides 15 general purpose registers (r0..r14) for integer, logical or address operands.

2.2.2 **Stack Pointer**

The stack pointer points at the next available 16 bit word on the call stack and is specified by the number 15 in register operand fields. The call stack is a downward growing stack. In case the stack is not used, the stack pointer can be utilized as general purpose register.

2.2.3 **Program Counter**

The program counter points at the instruction to be executed next.

2.2.4 Status Register

The fields of the status register are used to store the status of executed instructions.

reserved	exception	N	Z
----------	-----------	---	---

- The reserved bits of the status register are undefined and reserved for future use.
- The exception field is a 4 bit unsigned binary integer, which identifies an exception, that was raised by the last instruction.
- The N bit is 1 if the result of the last arithmetic instruction was negative.
- The Z bit is 1 if the result of the last arithmetic instruction was zero.

2.2.5 High Register

The HI register is used to store the high word of 32 bit values.

2.3 Exceptions

2.3.1 Access Exception (0)

The access exception is raised, if a memory location is accessed, that is not backed by physical memory.

2.3.2 Divide Exception (1)

The divide exception is raised if a divide instruction is executed with a divisor operand containing the value of zero.

2.4 pipeline

The pipeline consists of the following stages:

- Instruction fetch
- Instruction decode and Argument fetch
- Instruction execute
- Writeback

2.5 instruction families

table 1, instruction families

family	15 .. 12	11 .. 8	7 .. 4	3 .. 0
A	O O O O	r3 r3 r3 r3	r2 r2 r2 r2	r1 r1 r1 r1
B	O O O O	n n n n	n n n n	r1 r1 r1 r1
C	O O O O	O O O O	r2 r2 r2 r2	r1 r1 r1 r1
D	O O O O	O O O O	n n n n	r1 r1 r1 r1
E	O O O O	O O O O	O O O O	r1 r1 r1 r1
F	O O O O	O O O O	O O O O	O O O O
O .. Opcode rN .. register n .. immediate				

table 2, instruction set

family	instruction	opcode	meaning
A	add r1,r2,r3	0001	$r3 = r1 + r2$
	addu r1,r2,r3	0010	$r3 = r1 + r2$ (unsigned)
	sub r1,r2,r3	0011	$r3 = r1 - r2$
	subu r1,r2,r3	0100	$r3 = r1 - r2$ (unsigned)
	div r1,r2,r3	0101	$r3 = r1 / r2^1$
	divu r1,r2,r3	0110	$r3 = r1 / r2$ (unsigned) ¹
	mul r1,r2,r3	0111	$r3 = r1 * r2$
	mulu r1,r2,r3	1000	$r3 = r1 * r2$ (unsigned)
	and r1,r2,r3	1001	$r3 = r1 \wedge r2$
	or r1,r2,r3	1010	$r3 = r1 \vee r2$
xor r1,r2,r3	1011	$r3 = r1 \oplus r2$	
B	ldl r1, IMM	1100	$r1(7:0) = IMM[7:0]$
	ldh r1, IMM	1101	$r1(15:8) = IMM[7:0]$
	cij r1, IMM	1110	$r1 = PC^4 + IMM[7:0]$ (signed)
C	cmp r1,r2	0001 0001	$r1 \leq r2$, set flags
	cmpu r1, r2	0001 0010	$r1 \leq r2$, set flags (unsigned)
	not r1,r2	0001 0011	$r2 = \neg r1$
	shl r1,r2	0001 0100	$r1 = r1 \ll r2$
	shr r1,r2	0001 0101	$r1 = r1 \gg r2$
	shra r1,r2	0001 0110	$r1 = r1 \gg r2$ (sign extend)
	ldr r1,r2	0001 0111	$r1 = [r2]$
str r1,r2	0001 1000	$[r1] = r2$	
D	bset r1, IMM	0001 1001	$r1 = r1 \vee (1 \ll IMM[4:0])$
	bclr r1, IMM	0001 1010	$r1 = r1 \wedge \neg (1 \ll IMM[4:0])$
	btst r1, IMM	0001 1011	$Z = (r1 \gg IMM[4:0]) \wedge \neg 1$
	shli r1, IMM	0001 1100	$r1 = r1 \ll IMM[4:0]$
	shri r2, IMM	0001 1101	$r1 = r1 \gg IMM[4:0]$
E	beq r1	0000 0000 0001	$Z^2 == 1 \Rightarrow PC^3 = r1$
	bne r1	0000 0000 0010	$Z^2 == 0 \Rightarrow PC^3 = r1$
	ble r1	0000 0000 0011	$Z^2 == 1 \vee N^4 == 1 \Rightarrow PC^3 = r1$
	blt r1	0000 0000 0100	$N^4 == 1 \Rightarrow PC^3 = r1$
	bgt r1	0000 0000 0101	$Z^2 == 0 \wedge N^4 == 0 \Rightarrow PC^3 = r1$
	bge r1	0000 0000 0110	$Z^2 == 1 \vee (Z^2 == 0 \wedge N^4 == 0) \Rightarrow PC^3 = r1$
	jmp r1	0000 0000 0111	$PC^3 = r1$
	call r1	0000 0000 1000	$[sp] = PC^3, sp = sp - 2$
	push r1	0000 0000 1001	$[sp] = r1, sp = sp - 2$
	pop r1	0000 0000 1010	$r1 = [sp], sp = sp + 2$
mvhi r1	0000 0000 1011	$r1 = hi^5$	
F	ret	0000 0000 0001 0001	$PC^3 = [SP^6], sp = sp + 2$

X	A B	1111	not defined
	C D	0001 1110	not defined
	C D	0001 1111	not defined
	E	0000 0000 1100	not defined
	E	0000 0000 1101	not defined
	E	0000 0000 1110	not defined
	E	0000 0000 1111	not defined
	F	0000 0000 0001 0010	not defined
	F	0000 0000 0001 0011	not defined
	F	0000 0000 0001 0100	not defined
	F	0000 0000 0001 0101	not defined
	F	0000 0000 0001 0110	not defined
	F	0000 0000 0001 0111	not defined
	F	0000 0000 0001 1000	not defined
	F	0000 0000 0001 1001	not defined
	F	0000 0000 0001 1010	not defined
	F	0000 0000 0001 1011	not defined
	F	0000 0000 0001 1100	not defined, maybe bartender
	F	0000 0000 0001 1101	not defined, maybe kitchensink
	F	0000 0000 0001 1110	not defined, maybe make beer
F	0000 0000 0001 1111	not defined, maybe selfdestruct	

table 3, Status register

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
not used														N ⁴	Z ²

2.6 Assembler Instructions

The "mov r1,r2" instruction can be provided on assembler level, because it is equivalent to the machine instructions "and r1,r1,r2" and "or r1,r1,r2".

The "nop" instruction is equivalent to the machine instructions "and r1,r1,r1" and "or r1,r1,r1".

¹Division by zero triggers a hardware exception.

²Z: Zero flag, set when registers are equal.

³PC: Program counter, not directly addressable register.

⁴N: Flag indicating a value is smaller than the other in a compare.

⁵hi: Overflow register containing the overflow word of an arithmetic operation.

⁶SP: Stack pointer, if not used, the general purpose register R15.

References

- [1] David L. Weaver, Tom Germond, *The SPARC Architecture Manual, version 9*. PTR Prentice Hall, Englewood Cliffs, 1994.
- [2] SPARC International Inc., *The SPARC Architecture Manual, version 8*. SPARC International Inc., 535 Middlefield Road, Suite 210, 1991.
- [3] Sun Microsystems Inc., *SPARC Assembly Language Reference Manual*. Sun Microsystems Inc., 4150 Network Circle, Santa Clara, CA 95054, 2002.
- [4] wikipedia, GNU Free Documentation License, <http://en.wikipedia.org/wiki/SPARC>.
- [5] Atmel Corporation, <http://www.atmel.com>
- [6] AVR in wikipedia http://en.wikipedia.org/wiki/Atmel_AVR
- [7] AVR Freaks <http://www.avrfreaks.net>
- [8] IBM Systems/360 Principles of Operation, <http://portal.acm.org/citation.cfm?id=1102026>