Computer Engineering Department
Faculty of Engineering
Cairo University

Main Requirements

- Register Table
- ALU
- Control unit
- One level of memory

General Rules

- The processor required is a 32-bit RISC processor.
- The VHDL code must be functional (i.e. it allows functional simulation).
- Design your system using a Bottom-Up strategy, after all I want the system in the form of modules which interact with each other, using these modules in a hierarchical way to build your processor
- If you have any questions regarding VHDL, architectural design or project requirements, refer to your VLSI TA.
- Use test benches to test your design and to avoid forcing signals every time by hand
- Test vectors will be sent later to see how your processor will execute them.
- Any missing information will be assumed by you and hence you need to justify why you took these assumptions.

Memory Addressing

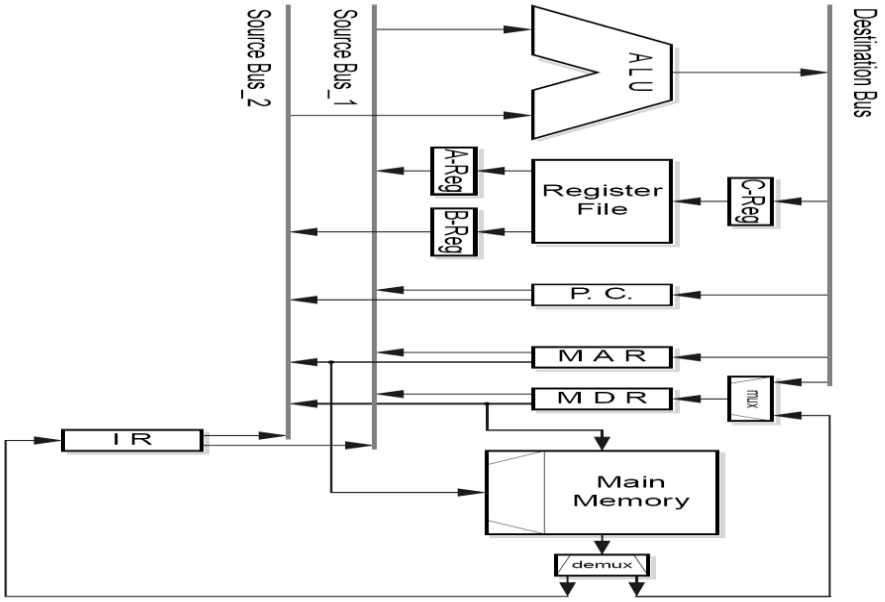
Memory is divided into words of 32 bits and is byte-addressed “Big Endian” mode, i.e., word addresses are multiples of 4. All memory addresses used will be relative addresses that start from ZERO.

Registers

Your processor should provide 32 general-purpose registers of 32 bits each named R0 - R31.

- R0 always contains zero. It can be used as a source operand whenever zero is needed, and stores to it have no effect.
- Branch instructions to subroutines implicitly use register R31 to store the return address
- R30 will be used as the SP.

The processor data path will be as shown in the figure below.



Instruction Set

Instruction Formats

There are three instruction formats in our processor: R-type, I-type, and J-type. All instruction formats must specify an opcode; however, the other information in the instruction varies by format. R-type (*register*) instructions specify three registers in the instruction - two source registers and one destination register. I-type (*immediate*) instructions specify one source register, one destination register, and a 16-bit immediate value that is sign-extended to 32 bits before it's used. J-type (*jump*) instructions consist of just the opcode and a 26 bit operand, which is used to calculate the destination address.

These three instruction formats are summarized in this table:

Format	Bits					
	31 26	25 21	20 16	15 11	10 6	5 0
R-type	0x0	Rs1	Rs2	Rd	unused	Opcode
I-type	Opcode	Rs1	Rd	Immediate		
J-type	Opcode	Value				

Instructions

Instruction	Description	Format	Opcode	Operation
ADD	Add	R	0x20	Rd = Rs1 + Rs2
ADDI	add immediate	I	0x08	Rd = Rs1 + extend(immediate)
AND	And	R	0x24	Rd = Rs1 & Rs2
ANDI	and immediate	I	0x0c	Rd = Rs1 & immediate
BEQZ	branch if equal to zero	I	0x04	PC += (Rs1 == 0 ? extend(immediate) : 0)
BNEZ	branch if not equal to zero	I	0x05	PC += (Rs1 != 0 ? extend(immediate) : 0)
J	jump	J	0x02	PC += extend(value)
JAL	jump and link	J	0x03	R31 = PC + 4 ; PC += extend(value)
JALR	jump and link register	I	0x13	R31 = PC + 4 ; PC = Rs1
JR	jump register	I	0x12	PC = Rs1
LHI	load high bits	I	0x0f	Rd = immediate << 16
LW	load word	I	0x23	Rd = MEM[Rs1 + extend(immediate)]
OR	Or	R	0x25	Rd = Rs1 | Rs2
ORI	or immediate	I	0x0d	Rd = Rs1 | immediate
SEQ	set if equal	R	0x28	Rd = (Rs1 == Rs2 ? 1 : 0)
SEQI	set if equal to immediate	I	0x18	Rd = (Rs1 == extend(immediate) ? 1 : 0)
SLE	set if less than or equal	R	0x2c	Rd = (Rs1 <= Rs2 ? 1 : 0)
SLEI	set if less than or equal to immediate	I	0x1c	Rd = (Rs1 <= extend(immediate) ? 1 : 0)
SLL	shift left logical	R	0x04	Rd = Rs1 << (Rs2 % 8)
SLLI	shift left logical immediate	I	0x14	Rd = Rs1 << (immediate % 8)
SLT	set if less than	R	0x2a	Rd = (Rs1 < Rs2 ? 1 : 0)
SLTI	set if less than immediate	I	0x1a	Rd = (Rs1 < extend(immediate) ? 1 : 0)
SNE	set if not equal	R	0x29	Rd = (Rs1 != Rs2 ? 1 : 0)
SNEI	set if not equal to immediate	I	0x19	Rd = (Rs1 != extend(immediate) ? 1 : 0)
SRA	shift right arithmetic	R	0x07	Same as SRL below*
SRAI	shift right arithmetic immediate	I	0x17	Same as SRLI below*
SRL	shift right logical	R	0x06	Rd = Rs1 >> (Rs2 % 8)
SRLI	shift right logical immediate	I	0x16	Rd = Rs1 >> (immediate % 8)
SUB	subtract	R	0x22	Rd = Rs1 - Rs2
SUBI	subtract immediate	I	0x0a	Rd = Rs1 - extend(immediate)
SW	store word	I	0x2b	MEM[Rs1 + extend(immediate)] = Rd
XOR	exclusive or	R	0x26	Rd = Rs1 ^ Rs2
XORI	exclusive or immediate	I	0x0e	Rd = Rs1 ^ immediate

*SRA and SRAI are arithmetic right shifts. This means that, instead of shifting in zeroes from the left, the sign bit of the operand is duplicated. SRL and SRA perform identically if Rs1 is positive. If Rs1 is negative (bit 31 == 1), 1's are shifted in from the left for SRA and SRAI.

RESET Signal

As with all processors, there must be some way of getting the CPU into a known state. This must be done after the CPU is "powered up", and can be done at other times to restore the CPU to a reasonable starting point. Our processor uses the RESET signal for this purpose.

- When the RESET signal is asserted (high), the CPU loads the program counter with 0.
- After RESET is deasserted, execution begins at location 0, which should probably be the address of the first instruction of a program to be executed.

Evaluation Criteria

Here are some guidelines to how the evaluation process will be going to give you some insight about what you need to care about when thinking about your design.

- Design and implementation will be compared relative to each other as a contest between several groups
- Teamwork (organization, efficient workload distribution, well documented reports, test benches and .do files for each component in your design)
- Comments
- Following a naming convention in your code

Deliverables

- Project Document, describing each module in your system and a small description on how it behaves for different inputs in different modes (I/P O/P relationship or in other words each module as a black box).
- The code for each group will be put in a folder named with the group name, all folders will be given to your class representative to be delivered to me on one CD.

Due Date and other issues

- Due date for the project will be Tuesday 24/4/12 9:00 AM (this due date is for both the CD delivered by your representative, in addition to a hardcopy of your project document in my mailbox delivered by each team leader in my mailbox)
- Discussion Date and Schedule will be announced later
- Meetings as well are to be scheduled with me in advance.
- Use your mail and don’t send to me a pm using unimasr.