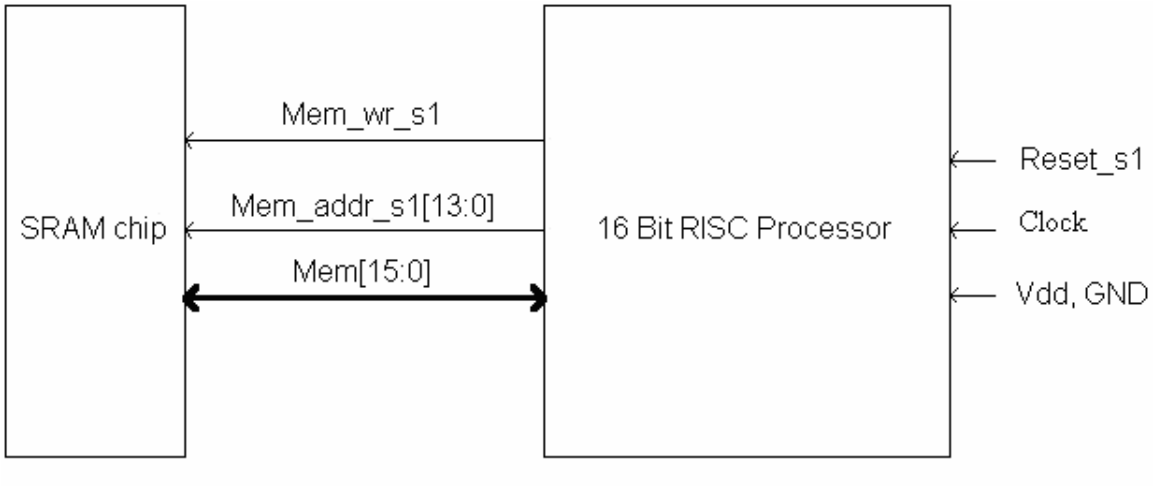

Fig.4 High Level Block Diagram that describes the external interface of the chip

4.1 Instruction Set Architecture (ISA)

The ISA of this processor consists of 16 instructions with a 4-bit fixed size operation code. The instruction words are 16-bits long. The following chart describes the instruction formats.

Operation	Opcode				Destination Reg				Source Reg				Target Reg			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD	0	0	0	0	Rd				Rs				Rt			
SUB	0	0	0	1	Rd				Rs				Rt			
AND	0	0	1	0	Rd				Rs				Rt			
OR	0	0	1	1	Rd				Rs				Rt			

XOR	0	1	0	0	Rd	Rs	Rt
NOT	0	1	0	1	Rd	Rs	
SLA	0	1	1	0	Rd	Rs	
SRA	0	1	1	1	Rd	Rs	
LI	1	0	0	0	Rd	Immediate	
LW	1	0	0	1	Rd	Rs	
SW	1	0	1	0		Rs	Rt
BIZ	1	0	1	1	Rs	Offset	
BNZ	1	1	0	0	Rs	Offset	
JAL	1	1	0	1	Rd	Offset	
JMP	1	1	1	0		Offset	
JR	1	1	1	1		Rs	

The Processor features five instruction classes:

1. Arithmetic (Two's Complement) ALU operation (2)

ADD: Rd = Rs + Rt

Operands A and B stored in register locations Rs and Rt are added and written to the destination register specified by Rd.

SUB: Rd = Rs - Rt

Operand B (Rt) is subtracted from Operand A (Rs) and written to Rd.

2. Logical ALU operation (6)

AND: Rd = Rs & Rt

Operand A (Rs) is bitwise anded with Operand B (Rt) and written into Rd.

OR: Rd = Rs | Rt

Operand A (Rs) is bitwise ored with Operand B (Rt) and written into Rd.

XOR: Rd = Rs ^ Rt

Operand A (Rs) is bitwise Xored with Operand B (Rt) and written into Rd.

NOT: Rd = ~Rs

Operand A (Rs) is bitwise inverted and written into Rd.

SLA: Rd = Rs << 1

Operand A (Rs) is arithmetically shifted to the left by one bit and written into Rd.

SRA: Rd = Rs >> 1

Operand A (Rs) is arithmetically shifted to the right by one bit and written into Rd. The MSB (sign bit) will be preserved for this operation.

3. Memory operations (3)

LI: Rd = 8-bit Sign extended Immediate

The 8-bit immediate in the Instruction word is sign-extended to 16-bits and written into the register specified by Rd.

LW: Rd = Mem[Rs]

The memory word specified by the address in register Rs is loaded into register Rd.

SW: Mem[Rs] = Rt

The data in register Rt is stored into the memory location specified by Rs.

4. Conditional Branch operations (2)

BIZ: PC = PC + 1 + Offset **if** Rs = 0

If all the bits in register Rs are zero than the current Program Count (PC + 1) is offset to PC + 1 + Offset. The count is offset from PC + 1 because it is incremented and stored during the Fetch cycle.

BNZ: PC = PC + 1 + Offset **if** Rs! = 0

If all the bits in register Rs are not zero than the current Program Count (PC + 1) is offset to PC + 1 + Offset.

5. Program Count Jump operations (3)

JAL: Rd = PC + 1 and PC = PC + 1 + Offset

Jump and Link instruction would write current Program Count in register Rd and offset the program count to PC + 1 + Offset

JMP: PC = PC + 1 + Offset

Unconditional jump instruction will offset the program count to PC + 1 + Offset.

JR: PC = Rs

Jump Return instruction will set the Program Count to the one previously stored in JAL.

FETCH INSTRUCTION

Part 1

- Retrieve instruction word from main memory
- Increment Program Counter and store in ALU Out

Part 2

- Write Incremented Program Count
- Load Operands into latches from Register File