CA Hardware Report

Memory:

We opted for a Harvard Architecture Model to decrease execution time of Instructions , we will be having two caches one for each memory.Our memory is considered a memory of an embedded system and that explains its small size of 8 KB, we have 12 address lines and each word is 2 Bytes,moreover, our memory is word addressable as the architecture would mainly focus on numerical operations and not text, this implementation would leave less bytes not in use when dealing with numbers.Having the architecture word addressable would also allow further jumps with same number of bits for offset (for example if we have an offset of 4 bits and we were using byte addressable memory, this offset would only address 8 words of our memory, however, when using word addressing this offset would be able to address 16 words) , Byte addressable memories favour Text operations as they store multiple ASCII Characters in the same word, so when retrieving a string we would require to access multiple bytes in the word separately instead of accessing the word as a whole.

ISA:

We chose a RISC design principle due to its simplicity in the data path design of our processor to be smaller and less complex.

Types of Registers

First Instruction type :

4 Bits OPCODE	4 Bits Source 1	4 Bits Source 2	4 bits Destination/ Offset

4 Bits OPCODE, 4 Bits dest.,4 bits source 1, 4 bits source 2
Second Instruction type :

4 bits OPCODE	12 Bits Address

4 Bits OPCODE 12 bits for Address
Third instruction type:

4 bits OPCODE	4 bits Destination/Source Register	8 bits Immediate

Name of Instruction	OPCODE	Formatting of the Instruction	Explanation
Branch if Equal (BIE)	0000	First Instruction Type	If Source 1 is Equal to Source 2 then PC wil been incremented by Offset
Jump Register (JR)	0001	First Instruction Type	Ignoring Source 2 and Destination and PC is set to word in Source 1 thus jumping
Branch Unconditionally (BUN)	0010	Second Instruction Type	PC is set to Address
Load from Memory (LD)	0011	First Instruction Type	Data from Adress(Source 1+ Offset) is loaded from memory into Source 2
Store in Memory (ST)	0100	First Instruction Type	Data from Source 2 is stored into memory Address(Source 1+ Offset)
AND	0101	First Instruction Type	An ALU operation (AND) is made on operands Source 1 Regisitr,Source 2 and is stored into Destination Register
OR	0110	First Instruction Type	An ALU operation (OR) is made on operands Source 1 Regisitr,Source 2 and is stored into Destination Register
ADD	0111	First Instruction Type	An ALU operation (Add) is made on operands Source 1

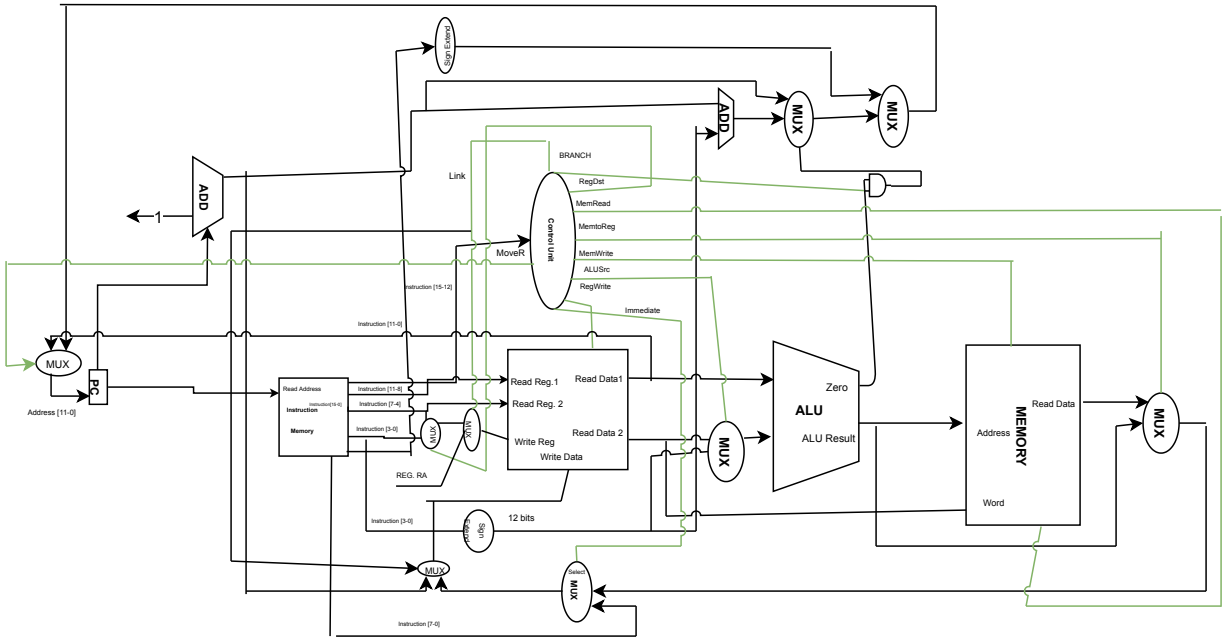
			Regisitr,Source 2 and is stored into Destination Register
Subtract (SUB)	1000	First Instruction Type	An ALU operation (Sub) is made on operands Source 1 Regisitr,Source 2 and is stored into Destination Register
Set less than (SLT)	1001	First Instruction Type	An ALU operation (Set Less Than) is made on operands Source 1 Regisitr,Source 2 and is stored into Destination Register
NOR	1010	First Instruction Type	An ALU operation (NOR) is made on operands Source 1 Regisitr,Source 2 and is stored into Destination Register
Move Immediate (MI)	1011	Third Instruction Type	Moves an Immediate into Destination Register
Shift Left (SL)	1100	First Instruction Type	An ALU operation (Shifting Left) is made on operands Source 1 Regisitr,Source 2 and is stored into Destination Register
Shift Right (SR)	1101	First Instruction Type	An ALU operation (Shifting Right) is made on operands Source 1 Regisitr,Source 2 and is stored into Destination Register
XOR	1110	First Instruction Type	An ALU operation (XOR) is made on

			operands Source 1 Regisitr,Source 2 and is stored into Destination Register
Call (CAL)	1111	Second Instruction	Branches to Address and saves the current PC into Return Register

Registers:

Register Name	Symbol	Number
Return Register	RR0	0
Return Register	RR1	1
Argument Register	A0	2
Argument Register	A1	3
Temp Reg.	T0	4
Temp Reg.	T1	5
Temp Reg.	T2	6
Temp Reg.	T3	7
Temp Reg.	T4	8
Saved Reg.	S0	9

Saved Reg.	S1	10
Saved Reg.	S2	11
Saved Reg.	S3	12
Saved Reg.	S4	13
Stack Pointer	SP	14
Return Address	RA	15


Sign Extend
ADD
MUX
MUX
ADD
1
Link
BRANCH
RegDst
MemRead
MemtoReg
MemWrite
ALUSrc
RegWrite
Control Unit
MoveR
Instruction [15-12]
Instruction [11-0]
Immediate
MUX
PC
Address [11-0]
Read Address
Instruction
Memory
Instruction [11-8]
Instruction [7-4]
Instruction [3-0]
MUX
MUX
Read Reg.1
Read Reg. 2
Write Reg
Write Data
Read Data1
Read Data 2
REG. RA
Instruction [3-0]
Sign Extend
12 bits
MUX
Select
MUX
Instruction [7-0]
Zero
ALU
ALU Result
MUX
Read Data
Address
MEMORY
Word
MUX