ARM Simulator

Meet Arora – 2016055

Gagandeep Singh - 2016037

The ARMSim Design:

Data Processing Instructions Implemented:

Condition	4 bits
Specification	2 bits
I	1 bit
Opcode	4 bits
S	1 bit
Rn	4 bits
Rd	4 bits
Operand2	12 bits

- ❏ SUB Rd, Rn, Rm | SUB Rd, Rn, #Imm
- ❏ ADD Rd, Rn, Rm | ADD Rd, Rn, #Imm
- ❏ ORR Rd, Rn, Rm | ORR Rd, Rn, #Imm
- ❏ CMP Rn, Rm | CMP Rn, #Imm
- ❏ EOR Rd, Rn, Rm | EOR Rd, Rn, #Imm

- ❑ AND Rd, Rn, Rm | AND Rd, Rn, #Imm
- ❑ ADC Rd, Rn, Rm | ORR Rd, Rn, #Imm
- ❑ MNV Rd, Rm | MNV RD, #Imm
- ❑ MOV Rd, Rm | MOV Rd, #Imm

Data Transfer Instructions Implemented:

Condition	4 bits
Specification	2 bits
Immediate	1 bit
Opcode	5 bits
Rn	4 bits
Rd	4 bits
Offset	12 bits

- ❑ LDR Rd, [Rn, #Imm] | LDR Rd, [Rn, Rm]
- ❑ STR Rd, [Rn, #Imm] | LDR Rd, [Rn, Rm]

Branch Instructions Implemented:

Condition	4 bits
Specification	2 bits
Opcode	2 bits
Offset	24 bits

- ❑ BNE
- ❑ BGT
- ❑ BLT
- ❑ BEQ
- ❑ BLE
- ❑ BAL or B
- ❑ BGE

Methodology:

We've created a **database** of the required instructions using a "**PresetInstruction**" class in a separate file which would contain all the required information such as opcode value, name of instruction, immediate value, etc.

FETCH:

The input file is read line by line. The two arguments are split. According to the given structure, the first argument would signify the instruction address and the second is the encoded instruction. The instruction is then written into the ins_MEM array. Finally, the output is printed as "FETCH:Fetch instruction $arg2 from address &arg1".

DECODE:

We first create a new Instruction object using the instruction encoding obtained from the .mem file. The class is created in such a way that all the fields of the instruction are calculated and stored in the Instruction object created. These fields are then used to match the instruction from the PresetInstruction database.

Then, we read the operand registers from the register file and save the values obtained in variables, say op1 and op2. After this we output in the following format-

"DECODE: Operation is XYZ, first operand is R2, second operand is R3, and destination register is R1"

"DECODE: Read registers R2=$op1, R3=$op2"

EXECUTE:

After the instruction has been decoded, the operation(if any) is performed on the operands obtained.

For example, if the operation is MOV r1, #10

We would first perform the operation, i.e. to move an immediate value of 10 to the register r1.

Then the result is printed in a format similar as follows tailored for each instruction.

“EXECUTE: Operation $val1 $val2 to $register_destination”

MEMORY:

Memory is accessed only during LDR/STR instructions.

Now, based on the instruction, if there is some operation to be performed on the memory file, the output will be as follows

“MEMORY: Written/Read to/from memory location”.

Otherwise

“MEMORY: No memory operation”.

WRITEBACK:

If the instruction requires a write-back to a register, it is performed using the result obtained from the previous stages and output in the following format “WRITEBACK: write $result to $register”.

References Used:

ARMv7 Manual