CO PROJECT

Topic: ARM SIMULATOR
Project Language: C

Theory:

Data processing instructions:

31-28	27-26	25	24-21	20	19-16	15-12	11-0
Condition	00	I	OpCode	s	Rn	Rd	Operand2

Rn: Operand 1 **Rd**: Destination register s: set condition codes
I: Immediate operand

Opcodes:
0000: AND (op1 AND op2)
0001: EOR (op1 EOR op2)
0010: SUB (op1-op2)
0011: RSB (op2-op1)
0100: ADD (op1+op2)
0101: ADC (add with carry)
0110: SBC (subtract with carry)
0111: RSC (reverse subtract with carry)
1000: TST (performs a bitwise AND operation)
1001: TEQ (performs a bitwise OR operation)
1010: CMP (compares op1 and p2- SUB)
1011: CMN (compares op1 and p2- ADD)
1100: ORR (op1 OR op2)
1101: MOV (mov op2 int op1)
1110: BIC (bit clear performs op1 AND NOT op2)
1111: MVN (mov NOT op2)

Instruction format for Load and Store:
(Load and Store use a different instruction format from above)

Condition	F	Opcode	Rn	Rd	Operand2
4 bits	2 bits	6 bits	4 bits	4 bits	12 bits

F: instruction format **Rn**: Operand 1 **Rd**: Destination register

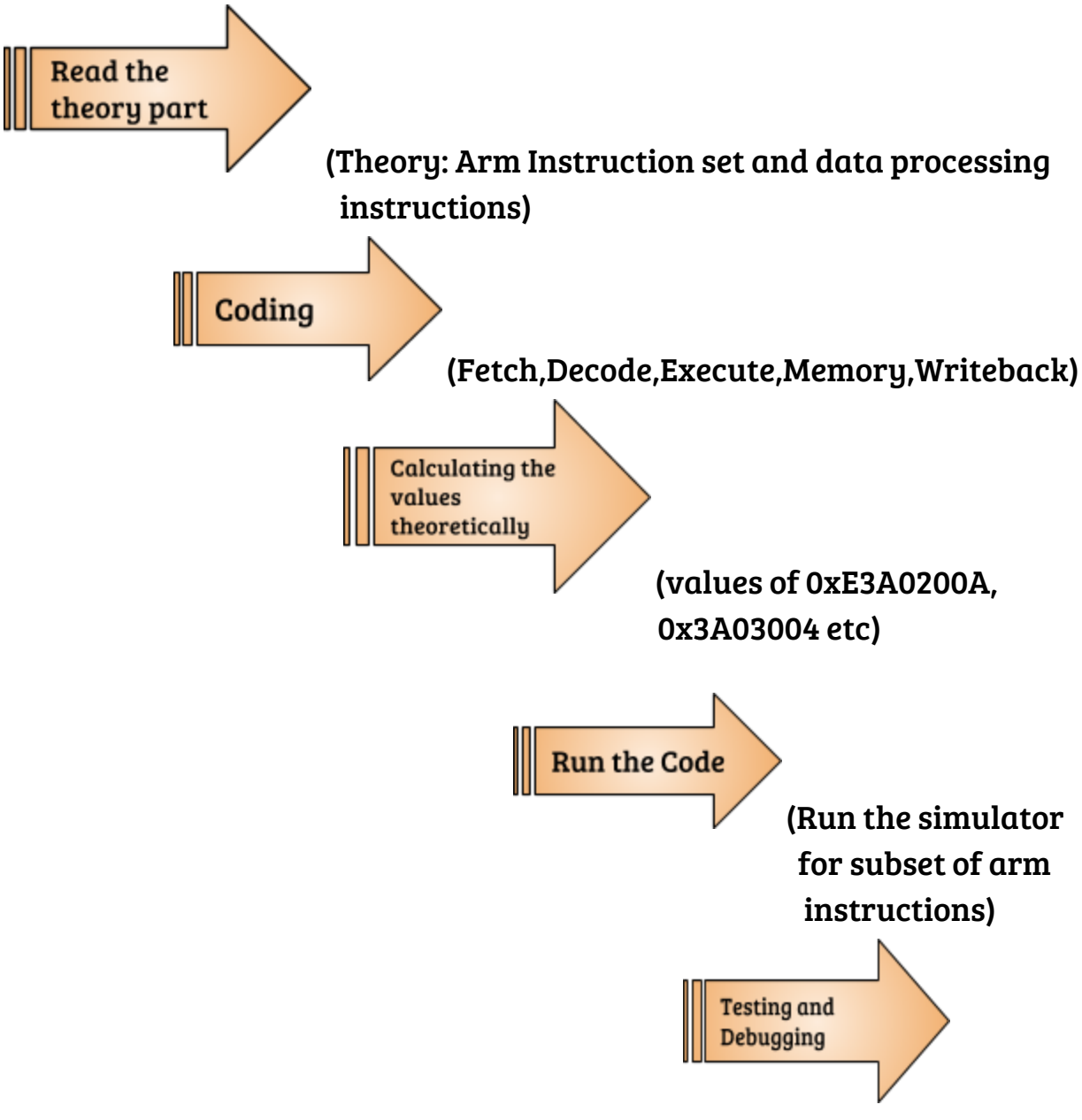
Instruction encoding:

Instruction	Format	Condition	F	Opcode	Rn	Rd	Operand 2
LDR	DT	14	1	24	reg	reg	address
STR	DT	14	1	25	reg	reg	address
ADD	DP	14	0	4	reg	reg	reg
SUB	DP	14	0	2	reg	reg	reg

LDR: load word
STR: store word
DT: data transfer instruction format
DP: data processing instruction format
reg: a register number between 0-15

Project Plan:

- We will use waterfall model in our project.
- Waterfall modelling: The linear consecutive structure for software development in which progress shown in downward direction.



Methodology:

After the header files, we will mention our function prototypes just to give an idea to the compiler about these functions.

Our main() function will consist of five function calls namely:

Fetch()
Decode()
Execute()
Write Back()
Memory()

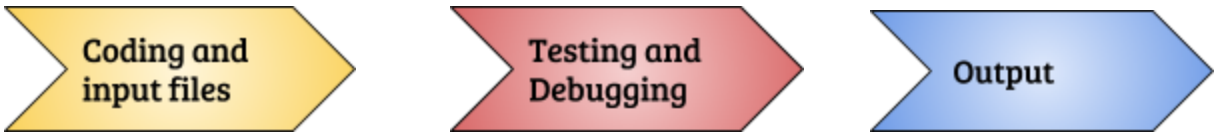
Apart from these function calls, our main() function will include the code to read the input file and static variables to ease the flow of control. Also there are two main steps which can be followed to perform the Simulator flow:

1. First memory is loaded with input memory file.
2. Simulator executes instruction one by one.
3. Simulator will run through an infinite loop till the instruction sequence "SWI 0x11" is encountered.

These functions will be defined outside the main() function. The function definition will be according to the following specifications:

- Fetch(): reads the data from instruction memory and updates the changes in the instruction register.
- Decode(): reads the instruction register, reads operand1, operand2 from register file and finally decides the operation to be performed in execute stage.
- Execute(): Perform all the arithmetic operations
- Memory(): Perform all memory operations like load word or store word.
- Write Back(): writes the final result to register.

Project Timeline:



(1-15th Nov)
Read the theory part and implementing it in our code.

(16th-25th Nov)
Theoretically calculating the values of instruction set provided in our input files.

(26th-28th Nov)
Functioning of ARM simulator on subset of arm instructions.

Team Members:

Richa Goswami (2016077)
Anushika Verma (2016015)
Alka Bharti (2016010)

Implemented Code Review:

Global Variables used:

- static unsigned int R[16];
  Registers used in the program are R0-R15
- static int N,C,V,Z;
- static unsigned char MEM[4000];
- static unsigned int word;
- static unsigned int operand1;
- static unsigned int operand2;
- static unsigned int destination;
  Register used to store the final result
- static unsigned int opcode;
  Opcode is used to determine the type of instructions
- static unsigned int format;
  format==0 (normal instructions)
  format==1 (memory instructions: LDR, STR)
  format==2(Branch instructions)

Functions of our code:

- void main(int argc, char** argv)
- void run_armsim();
  Fetch();
  Decode();
  Execute();
  Memory();
  write_back();
- void reset_proc();
  Used to set the reset values and reset all registers and memory content to zero

- void load_program_memory(char* file_name);
  It reads the input memory and populates the instruction memory

- void write_data_memory();
  Writes the data memory in "data_out.mem" file

- void swi_exit();
  Called when "SWI 0x11" is encountered.

- int read_word(char *mem, unsigned int address);

- void write_word(char *mem, unsigned int address, unsigned int data);

- void fetch();
  reads the data from instruction memory and updates the changes in the instruction register.

- void decode();
  Instructions used:
  AND, XOR, ADD, SUB, CMP, ORR, MOV, MVN
  Memory-LDR,STR
  Branch Instructions

- void execute();
  Perform all the arithmetic operations of the instruction used in the decode.

- void memory();
  LDR- read data from address
  STR- write data to address

- void write_back();
  Writes the final result(operand1) to destination register

Individual Contribution:

Anushika Verma(2016015):

- **Fetch()**
- **memory()**
- **write_back()**
- **read_word()**
- **write_word()**

Richa Goswami(2016077):

- **Execute()**
- **main()**
- **run_armsim()**
- **reset_proc()**
- **swi_exit()**

Alka Bharti(2016010):

- **Decode()**
- **load_program_memory()**
- **write_data_memory()**
- **Report**