ECEC 355 Drexel University, Winter 2021 3/13/2021

David Schmidt Damodhar Thota Mark Carson

## Project #4

<u>Task:</u> Divide the single-cycle core into five stages, represented by five new structures.

Implementing the five-stage architecture via structs was accomplished by creating a .h and .c file for each of the stages, fetch, decode, execute, memory access, and write back. Each .h file contained the struct type-defines and the associated function declarations. Each stage was executed by an init function that initialized the stage struct. In the init function space for the struct was malloc'd and struct members were assigned values based on the instruction, calls to included functions in the struct.{c,h} files, and values from the previous stages structs that were passed into the init. The execution of each instruction was facilitated in the Core.c tickFunc function. Previous versions of the simulation executed the datapath in tickFunc, the new implementation reduced tickFunc to five initializations of stage structs that call the associated struct init function. The tickFunc function also contained a few required lines of code to facilitate the program execution. The general flow of tickFunc is given below. The biggest challenge in the project was properly setting up all the necessary #include statements necessary to execute functions and struct access between different files.

Figure 1 shows the initialized register and data memory values as given in the project guidance document. Figure 2 gives an example of the debugging output generated for the ld instruction at PC = 12. Both figures are given on page 2.

## 5-stage execution in tickFunc, code snippet:

```
IFe *fetch = initIFe(core);
ID *id = initID(fetch, core);
EX *ex = initEX(id);
MEM *mem = initMEM(id, ex, core);
WB *wb = initWB(id, ex, mem, core);
```

```
*** Simulation is finished. ***
initialized reg[] & mem[] values
reg[1] = 0
reg[2] = 10
reg[3] = -15
reg[4] = 20
reg[5] = 30
reg[6] = -35
mem[40] = -63
mem[48] = 63
resultant reg[] & mem[] values
reg[10] = -63
reg[11] = 25
reg[12] = 5
reg[13] = 63
reg[14] = -5
[mbc93@xunil-03 project 4]$
```

Figure 1: Initialized register and data memory values as given in the project guidance document.

```
Instruction 0x0300B683 at PC: = 12
*** ID STRUCT: I-TYPE ***
opcode: 3
rd: 13
funct3: 3
rs1: 1
imm: 48
ALUop: 0
ALU_in1: 0
ALU in2 (imm): 48
printed in initEX before ALU MUX() call:
select = 1, MUXin1 = 20, MUXin2 = 48
printed in initEX after ALU_MUX() call:
ALU_in1 = 0, ALUin2 = 48
printed in initEX after ALU() call:
ALU_result = 48, zero = 0
*** PRINTED IN WB STRUCT ***
reg[13] = mem[48] = 63
```

Figure 2: Example of the debugging output generated for the ld instruction at PC = 12.