ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION PROJECT REPORT

PROJECT NO	:	2
GROUP NO	:	G63

GROUP MEMBERS:

150210728	:	ERCE GÜLMEZ
150200717	:	KÖKSAL FİŞENGİ
150190058	:	MEHMET EMRE TOPDAL

SPRING 2023

Contents

1 INTRODUCTION

In this project, we implemented the basic operations that a simple computer executes. We used the components that we designed in project 1 and added some new modules that helps us to give the right signals to the components of the computer. By giving the right signals to those components at the right time, we are expecting to obtain right result for each operation. Our implementation starting from the fetch&decode cycle is explained in detail on the next section.

2 PROJECT DETAILS

2.1 FETCH AND DECODE

The Reset condition checks if a reset signal is asserted. If true, the following operations are performed:

- sc is set to 1 to continue to make operations in the next cycle.
- Mem WR is set to 0, indicating a read operation from memory.
- Mem CS is set to 1, disabling the memory chip.
- RF RegSel and ARF RegSel are set to 4'b1111, enabling all registers in the register file and address register file, respectively.
- IR Enable is set to 1, enabling the instruction register.
- RF FunSel, ARF FunSel, and IR FunSel are set to 2'b00, which clears the enabled registers in the register file, address register file, and instruction register. This operation prevents override.

The T0 condition checks if a signal T0 is asserted. If true, the following operations are performed:

- ARF OutASel is assigned 2'b11, indicating that the OutA is connected to the address register.
- MuxBSel is assigned 2'b11, indicating that address register file takes ARF OutA as an input from mux b.
- ARF RegSel is set to 4'b0101, selecting address register and program counter past in the address register file.

- ARF FunSel is set to 2'b01, indicating that load operation will be executed in enabled registers(AR, PC PREV) in the address register file

The T1 condition checks if a signal T1 is asserted. If true, the following operations are performed:

- IR LH is assigned 0 and IR FunSel is set to 2'b01 to load first eight bit to instruction register.
- IR Enable is set to 1, enabling the instruction register.
- Mem WR and Mem CS are both set to 0, enabling memory access for reading.
- RF RegSel is set to 4'b0000 and RF TSel is set to 4'b0000 to disable registers in the register file.
- ARF OutBSel is assigned 2'b00, indicating that the OutB is connected to the address register.
- ARF RegSel is set to 4'b1001, selecting program counter and program counter past in the address register file..
- ARF FunSel is set to 2'b11, indicating that increment operation will be executed in enabled registers(PC, PC PREV) in the address register file

The T2 condition checks if a signal T2 is asserted. If true, the following operations are performed:

- IR LH is assigned 1 and IR FunSel is set to 2'b01 to load last eight bit to instruction register.
- IR Enable is set to 1, enabling the instruction register.
- Mem WR and Mem CS are both set to 0, enabling memory access for reading.
- RF RegSel is set to 4'b0000 and RF TSel is set to 4'b0000 to disable registers in the register file.
- ARF OutBSel is assigned 2'b10, indicating that the OutB is connected to the program counter previous.
- ARF RegSel is set to 4'b1000, selecting program counter in the address register file.
- ARF FunSel is set to 2'b11, indicating that increment operation will be executed in enabled registers(PC) in the address register file

2.2 OPERATIONS

2.2.1 MOV/NOT/LSL/LSR

All there four operations' code are very similar such that only ALU FunSel will change. Other control signals which we consider will be same. Firstly we disable IR by giving IR-Enable=0 and Mem-CS = 1. Since these operations take just 1 clock cycle, we will use ARF and RF in load mode (RF-FunSel = ARf-FunSel = 01) and arrange RSel signals to specify destination register. We will specify RSel signals by using DestSelectRF and DestSelectARF values by decoding first two bits of SREG1 and DESTREG

- if SREG1 is in RF, we will give MuxCSel = 0 and RF-Out1Sel = 1'b1,SREG1[1:0] to give SREG1 through MuxC into ALU
- if SREG1 is in ARF, we will give MuxCSel = 1 and ARF-Out1Sel = SREG1[1:0] to give SREG1 through MuxC into ALU
- if DESTREG is in RF, we will give MuxASel = 00 and RF-RegSel = DestSelectRF to give ALUOut into RF through MuxA. We disable ARF to not encounter data loss
- if DESTREG is in ARF, we will give MuxBSel = 00 and ARF-RegSel = DestSelectARF to give ALUOut into RF through MuxB. We disable RF to not encounter data loss

2.2.2 INC/DEC

Increment and Decrement operations have 2 clock cycle times. In first cycle, we MOV SREG to DESTREG. At secand clock cycle, we increment the DESTREG. As we point out, first clock cycle is same as MOV operation at previous part, therefore we just explain how second clock operations work

At second clock cycle, differance between INC and DEC operations is just registers' FunSel. Memory and IR will are disabled. ARF and RF are used in increment(11) or decrement(10) mode. We will specify RSels to determine which registers will be enabled. We will specify RSel signals by using DestSelectRF and DestSelectARF values by decoding first two bits of SREG1 and DESTREG. Of course; if DEST register in ARF, RF will be disabled. If DEST register in RF, ARF will be disabled.

2.2.3 AND/OR/ADD/SUB

Alu-Funsel decide which operation will be executed. First we disable instruction register.Mem-WR is set to 0, indicating a read operation from memory. Mem CS is set to 1, disabling the memory chip. If IROuts eleventh bit is equal to 0, that means load output into the register file and these operation will be executed.

- First we set RF-Funsel 01 to load result to register in register file that is selected by DSTREG. For selecting registers, we write A decoder that gives output as a DestSelectRF. We set ARF-Regsel 0000 to disable register in adress register file. We set MuxASel 00 to select OutAlu as the input of register file.If IROuts seventh and third bit is equal to 0 ,that means SREG1 and SREG2 comes from register file and these operations will be executed.
  - We set MuxCSel 0 to select register file. Then we set RF-Out1Sel to 1 for left most bit and we set RF-Out1Sel to IROut[5:4] for remaining bits. At the end for this condition we set RF-Out2Sel to 1 for left most bit and we set RF-Out2Sel to IROut[1:0] for remaining bits.
- If IROuts seventh bit is equal to 1 and third bit is equal to 0 ,that means SREG1 comes from adress register file , SREG2 comes from register files and these operations will be executed
  - We set MuxCSel 1 to select adress register file. Then we set ARF-OutASel to IROut[5:4] . At the end for this condition we set RF-Out2Sel to 1 for left most bit and we set RF-Out2Sel to IROut[1:0] for remaining bits.
- If IROuts seventh bit is equal to 0 and third bit is equal to 1 ,that means SREG1 comes from register file , SREG2 comes from adress register files and these operations will be executed
  - We set MuxCSel 1 to select adress register file. Then we set ARF-OutASel to IROut[1:0] . At the end for this condition we set RF-Out2Sel to 1 for left most bit and we set RF-Out2Sel to IROut[1:0] for remaining bits.

If IROuts eleventh bit is not equal to 0, that means load output into the address register file and these operation will be executed.

- First we set ARF-Funsel 01 to load result to register in adress register file that is selected by DSTREG. For selecting registers, we write a decoder that gives output as a DestSelectARF. We set RF-Regsel 0000 to disable register in register file. We set MuxBSel 00 to select OutAlu as the input of adress register file.If IROuts seventh and third bit is equal to 0 ,that means SREG1 and SREG2 comes from register file and these operations will be executed.

  - We set MuxCSel 0 to select register file. Then we set RF-Out1Sel to 1 for left most bit and we set RF-Out1Sel to IROut[5:4] for remaining bits. At the end for this condition we set RF-Out2Sel to 1 for left most bit and we set RF-Out2Sel to IROut[1:0] for remaining bits.

- If IROuts seventh bit is equal to 1 and third bit is equal to 0 ,that means SREG1 comes from adress register file , SREG2 comes from register files and these operations will be executed

  - We set MuxCSel 1 to select adress register file. Then we set ARF-OutASel to IROut[5:4] . At the end for this condition we set RF-Out2Sel to 1 for left most bit and we set RF-Out2Sel to IROut[1:0] for remaining bits.

- If IROuts seventh bit is equal to 0 and third bit is equal to 1 ,that means SREG1 comes from register file , SREG2 comes from adress register files and these operations will be executed

  - We set MuxCSel 1 to select adress register file. Then we set ARF-OutASel to IROut[1:0] . At the end for this condition we set RF-Out2Sel to 1 for left most bit and we set RF-Out2Sel to IROut[1:0] for remaining bits.

At the end, we reset the counter since this operation ends here.

Since these operations are very similar, only the ALU-Funsel is changed during the implementation. So, there is no need to explain each of them in detail, the procedures are exactly the same.

2.2.4 BRA

This operation takes the VALUE which is actually present in IR[7:0] and loads that value to the PC. In order to do it, we set ARF-funsel to 01 to load, MuxBsel to 10 to select IR[7:0] as its output and ARF-regsel to 1000 to enable only PC. At the end, we reset the counter since this operation ends here.

2.2.5 BNE

This operation takes the VALUE which is actually present in IR[7:0] and loads that value to the PC if Z's value is equal to 0. In order to do it, we set ARF-funsel to 01 to load, MuxBsel to 10 to select IR[7:0] as its output and ARF-regsel to 1000 to enable only PC. At the end, we reset the counter since this operation ends here. We put these assignments in an if block that checks if ALUOutFlag[3] (which is Z) is equal to 0 or not.

2.2.6 LD

This operation takes the VALUE which is actually present in IR[7:0] and loads that value to the desired register in RF. If the addresing mode is IM, we set MuxAsel to 10 to select IR[7:0] as its output, RF-funsel to 01 to make it load and since one of our decoders sets RF-regsel according to the regsel information taken from the instruction, this operation becomes completed.At the end, we reset the counter since this operation ends here.

If the addressing mode is D, the operation takes 2 clock cycles. For the first cycle, we set MuxBsel to 10 to select IR[7:0] as its output, ARF-regsel to 0100 to enable only AR and ARF-funsel to 01 to load. On the second clock cycle, we set ARF-OutBsel to 00 to give memory the address which is equal to AR's value. We read the value on that address from the memory and load it to the desired register in RF by setting the components just like IM mode. At the end, we reset the counter since this operation ends here.

2.2.7 ST

This operation writes the value of the RX into the memory. We set RF-Out2sel to 1'b1,regsel to give the RX's value to ALU, ALU-Funsel to 0001 to let ALU give that value directly as its output, ARF-OutBsel to 00 to give AR's value as the address and at last we write the OutALU to that address cell on memory. At the end, we reset the counter since this operation ends here.

2.2.8 PUL / PSH

This operation takes 2 clock cycles. The first cycle increments the SP and the second cycle loads the value which is written on the address provided by SP to the desired register in RF. For the first cycle, we set ARF-Regsel to 0010 to select SP, ARF-Funsel to 11 to increment and ARF-OutBsel to 01 to give SP's value as the address to the memory. The second cycle sets RF-Funsel to 01 to load and MuxAsel to 01 to select MemoryOut as the input of the RF. Since one of our decoders sets RF-regsel according to the regsel information taken from the instruction, this operation becomes completed. At the end, we reset the counter since this operation ends here.

The PSH operation is the exact opposite of to the PUL operation. We first write the value of the RX to M[SP] and then decrement the SP on the second clock cycle. Since the process is very similar to PUL there is no need to explain it in detail.

3 DISCUSSION

In the first assignment, we tried to design a basic computer by giving control signals to the system we implemented. We set these control signals in a separate module in Opcodes and connected Opcodes and ALUSystem module with cables to enable the two systems to communicate with each other.

We took the IROut value from ALUSystem and divided the instructions into the necessary parts (opcode, SREG, Address). In this way, we set the control signals in the required time variables (T). Then we transmitted these control signals to ALUSystem.

4 CONCLUSION

The most difficult part of the assignment was to understand the logic of the assignment. At first, we didn't know which modules to divide the system into, so we had a hard time building a general structure. But once we solved this problem, we wrote the program quickly. We learned how to use Verilog in more detail and got a deeper understanding of systems similar to Morris Mano's computer.