MIDDLE EAST TECHNICAL UNIVERSITY

NORTHERN CYPRUS CAMPUS

MIDDLE EAST TECHNICAL UNIVERSITY NORTHERN CYPRUS CAMPUS

ELECTRICAL AND ELECTRONICS ENGINEERING PROGRAM

EEE 446 Computer Architecture

Lab Module 3

CPU ISA & DATAPATH DESIGN

Preliminary Report

Name, Surname	Doğukan Fikri Arat
Student ID	2079648
Name, Surname	Saad Yousaf
Student ID	2246536

MULTI-8 ISA

ISA Overview

An ISA is designed to accommodate all the operations and the functions the CPU will be required to perform. For our MULTI-8 ISA and CPU design the required target is assessed from the benchmark provided in lab manual section 6.2. After assessing the requirements for this benchmark and using the multicycle CPU design approach we developed our Instruction formats and list of instructions that the CPU needs to perform for the benchmark. Mostly we have implemented an ISA similar to the MIPS ISA. The provided incomplete datapath has been modified to accommodate our ISA and multicycle CPU design.

Instruction formats:

R-TYPE				
4-bit	3-bit	3-bit	3-bit	3-bit
OPCODE (Op)	DESTINATION (Rd)	SOURCE_1 (Rs)	SOURCE_2 (Rt)	OAP
I-TYPE				
4-bit	3-bit	3-bit	6-bit	
OPCODE (Op)	DESTINATION (Rd)	SOURCE_1 (Rs)	CONSTANT (C)	
J-TYPE				
4-bit	12-bit			
OPCODE (Op)	CONSTANT (C)			

Table 1: Different instruction types supported by MULTI-8

- R-type
  All R-type instructions have a 4-bit opcode, which is 0000 for all R-type instructions. The instruction itself is determined by the 3 OAP bits, which means we have 8 R-type instructions. There are two source registers Rs and Rt, 3-bit address each, and one 3-bit destination register Rd. R-type instructions are arithmetic-logic operations and some of them have immediate versions as well which are defined under I-type instructions.

- I-type
  All I-type instructions have a 4-bit OPCODE which is used to identify the instruction as well in the decode stage. The is one destination (Rd) and one source (Rs) register, both having 3-bit addresses. The immediate is a 6-bit constant. This can be used with I-type instructions after sign-extension to 8-bit, since all operations are using 8-bits.
- J-type
  We currently have 1 J-type instruction, which is Jump instruction. It has a 4-bit OPCODE and a 12-bit constant which is the destination address for the jump target.

ISA Design Decisions:

All instruction types have a 4-bit OPCODE which is used to decode which type of instruction it is and identify the instruction in case of I or J type. In case of R-type, all the OPCODEs are same (i.e. 0000) and the instruction is identified using the 3 OAP bits. We chose the OPCODE to be 4-bits long by assessing how many instructions we are expected to have in total. Initially we have 19 instructions so 4-bits are enough to address them. And, since all R-type instructions have the same OPCODE we can have 8 R-type instructions with OPCODE 0000, but they are identified by the 3 OAP bits. Putting it together we still have a room for expansion for 5 instructions. These extra instructions can be added as I or J type.

The destination (Rd) and source (Rs and Rt) registers in R and I-type instructions have been dedicated 3-bits for address. 3-bits are chosen for this purpose as our register file is 8X8. We have 8 registers that we need to address so 2^3 registers. Therefore, we decided to use 3-bits for register addressing.

After allocating the bits required for OPCODE, source and destination in I-type instructions we were left with 6-bits. We allocated all these 6-bits for the immediate constant to be used in immediate operations. Therefore, our immediate operations are limited by only 6-bit constants.

For j-type instructions we allocated 4-bits for OPCODE and the remaining 12-bits for Constant. The constant can directly be used as the jump target location address. Since our instruction memory has 10-bits for addressing its 2 KB size, the 12- bit constant for J-type instructions can address all this memory, in fact the 12-bit constant can be used to address instruction memory for up to 4 KB (12-bit address)

MULTI-8 ISA Instructions:

The various instructions listed in Table 2 below are based on the provided benchmark for this CPU design. Looking at the n-long array computation we require basic instructions such as addition, subtraction, shift operations, branch, load and store. The multiplication and division instructions are required for the multiplication and division benchmarks. The jump instruction will be useful in the infinite loop benchmark. All the previously mentioned instructions combined with some other instructions are required or the text parser benchmark.

Instruction	Type	Operands	Description	Operation	Opcode	OAP
ADD	R	Rd, Rs, Rt, Cin	Add two registers and a carry (Cin)	Rd <= Rs + Rt + Cin	0	0
ADDI	I	Rd, Rs, C	Add a register and a constant (C)	Rd <= Rs + C	1	X
SUB	R	Rd, Rs, Rt, Cin	Subtract two registers and a carry (Cin)	Rd <= Rs - Rt - Cin	0	1
AND	R	Rd, Rs, Rt	Logical AND two registers	Rd <= Rs ^ Rt	0	4
ANDI	I	Rd, Rs, C	Logical AND two registers and a constant (C)	Rd <= Rs ^ C	2	X
OR	R	Rd, Rs, Rt	Logical OR two registers	Rd <= Rs ^{v} Rt	0	3
DMADDR	J	C	Set most significant 2-bits of Data Memory Address to least significant 2-bits of constant (C)	DM_address[8:9] <= C [1:0]	3	X
XOR	R	Rd, Rs, Rt	Logical XOR two registers	Rd <= Rs $\oplus$ Rt	0	6
SLT	R	Rd, Rs, Rt	If Rs<Rt, Rd=1, else Rd=0.	Rs-Rt, Rd <= N (zero flag)	0	5
MUL	R	Rd, Rs, Rt	Multiply two registers	[Rd,Rd+1] <= Rs * Rt	0	2
DIV	R	Rd, Rs, Rt	Divide two registers	Rd <= Rs / Rt	0	7
SLL	I	Rd, Rs, C	Logical shift left Rs and save into Rd with constant amount (C)	Rd <= Rs << C	4	X
SRL	I	Rd, Rs, C	Logical shift right Rs and save into Rd with constant amount (C)	Rd <= Rs >> C	5	X
SRA	I	Rd, Rs, C	Arithmetic shift right Rs and save into Rd with constant amount (C)	Rd <= Rs >> C	6	X
LW	I	Rd, Rs, C	Load 16-bit word from data memory	[Rd,Rd+1] <= MEM [Rs + C]	7	X
LWU	I	Rd, Rs, C	Load upper byte from data memory location	[Rd, -] <= MEM [Rs + C]	8	X
LWL	I	Rd, Rs, C	Load lower byte from data memory location	[-, Rd] <= MEM [Rs + C]	9	X
SW	I	Rd, Rs, C	Store 16-bit word to data memory	MEM [Rs + C] <= [Rd,Rd+1]	10	X
SWU	I	Rd, Rs, C	Store 8-bit data to upper byte of data memory location	MEM [Rs + C] <= [Rd, -]	11	X

Table 2: List of Instructions

Instruction	Type	Operands	Description	Operation	Opcode	OAP
SWL	I	Rd, Rs, C	Store 8-bit data to lower byte of data memory location	MEM [Rs + C] <= [-,Rd]	12	X
BREQ	I	Rd, Rs, D	Branch if Rd equals Rs	if Rs-Rd=0, i.e. Z=1, PC <= PC + D	13	X
BRNE	I	Rd, Rs, D	Branch if Rd not equal to Rs	if Rs-Rd!=0, i.e. Z=0, PC <= PC + D	14	X
JUMP	J	D	Jump to address location (D)	PC <= PC+D	15	X
**** D is the jump or branch Destination Address**						

Table 2: List of Instructions

Complete Datapath Design

For Shift op, Data1 directly go to control unit. to control Shift registers

R type ✓
I type ✓
J type ✗

Addi $R_1, R_0, 15$
addi $R_2, R_0, 15$
ld1 $R_1, R_2(C)$
R_d R_s
ld D $R_1, R_2(C)$
sw1 $R_1, R_2(C)$