
HARDWARE
DESIGN
PROJECT

1. The assignment Aim

Aim:

The overarching aim of this project is to design, implement, and evaluate a Single Cycle MIPS Processor using VHDL. This will provide a practical understanding of processor architecture and digital system design using VHDL.

Objectives:

1. ***Understanding the MIPS Instruction Set:***

   This involves a comprehensive study of the MIPS instruction set architecture (ISA). The ISA defines the supported data types, the registers, the instruction format, and the addressing modes, as well as the native operations such as arithmetic, bit manipulation, or memory access operations. Understanding these elements is crucial for the design of the processor.

2. ***Designing the Processor Architecture:***

   This involves designing the data path and control units of the processor. The data path includes components like the ALU (Arithmetic Logic Unit), registers, and buses, while the control unit controls the flow of data within the CPU. The design should be efficient and optimized for the MIPS instruction set.

3. ***Implementing the Processor in VHDL:***

   This involves translating the processor design into VHDL code. VHDL is a hardware description language used in electronic design automation to describe digital and mixed-signal systems. The implementation should be modular, with separate VHDL files for different components of the processor. This will make the code easier to understand, test, and debug.

4. ***Testing and Evaluation:***

   This involves verifying the correctness of the VHDL implementation by running a set of test instructions on the processor. The performance of the processor should also be evaluated in terms of criteria like clock cycles per instruction and resource utilization. Any discrepancies or inefficiencies should be identified and addressed.

5. ***Documentation:***

   All stages of the project are thoroughly documented. This includes the design rationale, the VHDL code, test procedures and results, and performance evaluation. The documentation serves as a record of the project and can be useful for future reference or improvement.

By achieving these objectives, we aim to gain a comprehensive understanding of processor design and the practical application of VHDL in digital system design.

2. The Problem Solution

2. 1. Inputs and Output

Instructions Format:

Operation	Opcode	Funct	Op (Enters ALU)
Addition	000000	001100	001
Subtraction	000000	110011	011
Load	101010		
Store	010101		
Branch Equal	100000		
Branch Not Equal	000001		
And	000000	011011	111
Or	000000	110101	010

Data Before Running Instructions:

Reg Name	Reg Index	Reg Value	DM Address	DM Index	DM Value
`$zero`	0	1	4	0	526
`$at`	1	44	5	1	0
`$v0`	2	91	6	2	60
`$v1`	3	136	7	3	0
`$a0`	4	1	8	4	0
`$a1`	5	0	9	5	0

$a2	6	381	10	6	0
$a3	7	3	11	7	0
$t0	8	559	12	8	0
$t1	9	648	13	9	0
$t2	10	739	14	10	0
$t3	11	2	15	11	0
$t4	12	921	16	12	0
$t5	13	1012	17	13	0
$t6	14	1103	18	14	0
$t7	15	1196	19	15	0
$s0	16	1289			
$s1	17	1382			
$s2	18	1475			
$s3	19	1568			
$s4	20	1661			
$s5	21	1754			
$s6	22	1847			
$s7	23	1940			
$t8	24	2033			
$t9	25	2126			
$k0	26	2219			
$k1	27	2310			
$gp	28	2403			
$sp	29	2496			
$fp	30	2589			
$ra	31	2682			

Instructions:

1- add $s0, $a2, $zero

Inst.	opcode	$a2	$zero	$s0	shamt	funct
Code	000000	00110	00000	10000	00000	001100
Index	R-Type	6	0	16		Add
Val Before		381	1	1289		
Val After		381	1	382		

2- beq $sp, $t2, 0 False

Inst.	opcode	$sp	$t2	offset
Code	100000	11101	01010	00000 00000 000000
Index	Beq	29	10	Next Inst.
Val Before		2496	739	
Val After		2496	739	

3- add $t6, $s3, $gp Will Not BE Executed

Inst.	opcode	$s3	$gp	$t6	shamt	funct
Code	000000	10011	11100	01110	00000	001100
Index	R-Type	19	28	14	0	Add
Val Before		1568	2403	1103		
Val After		1568	2403	1103		

4- bne $k1, $t8, 0 True

Inst.	opcode	$k1	$t8	offset
Code	000001	11011	11000	00000 00000 000000
Index	Bne	27	24	Next Inst.
Val Before		2310	2033	
Val After		2310	2033	

5- sub $fp, $ra, $t5 Will BE Executed

Inst.	opcode	$ra	$t5	$fp	shamt	funct
Code	000000	11111	01101	11110	00000	110011
Index	R-Type	31	13	30		Sub
Val Before		2682	1012	2589		
Val After		2682	1012	1670		

6- beq $t1, $t9, 0 False

Inst.	opcode	$t1	$t9	offset
Code	100000	01001	11001	00000 00000 000000
Index	Beq	9	25	Next Inst.
Val Before		648	2126	
Val After		648	2126	

7- sub $k0, $t4, $v1 Will Not BE Executed

Inst.	opcode	$t4	$v1	$k0	shamt	funct
Code	000000	01100	00011	11010	00000	110011
Index	R-Type	12	3	26		Sub
Val Before		921	136	2219		
Val After		921	136	2219		

8- bne $s7, $t0, 0 True

Inst.	opcode	$s7	$t0	offset
Code	000001	10111	01000	00000 00000 000000
Index	Bne	23	8	Next Inst.
Val Before		1940	559	
Val After		1940	559	

9- lw $at, 4($a1) Will BE Executed

Inst.	opcode	$at	$a1	increment
Code	101010	00101	00001	00000 00000 000100
Index	Load	1	5	
Val Before		44	0	4
Val After		526	0	
DM Address	4($a1) = 4 + 0 = 4			
DM Index	0			
DM Val Before	526			
DM Val After	526			

10- sw $t7, 4($a0)

Inst.	opcode	$t7	$a0	increment
Code	Store	00100	01111	00000 00000 000100
Index		15	4	
Val Before		1196	1	4
Val After		1196	1	
DM Address	4($a0) = 4 + 1 = 5			
DM Index	1			
DM Val Before	0			
DM Val After	1196			

11- lw $v0, 4($t3)

Inst.	opcode	$v0	$t3	increment
Code	101010	01011	10010	00000 00000 000100
Index	Load	2	11	
Val Before		91	2	4
Val After		60	2	
DM Address	4($t3) = 4 + 2 = 6			
DM Index	2			
DM Val Before	60			
DM Val After	60			

12- sw $s2, 4($a3)

Inst.	opcode	$s2	$a3	increment
Code	010101	00111	10010	00000 00000 000100
Index	Store	18	7	
Val Before		1475	3	4
Val After		1475		
DM Address	4($a3) = 4 + 3 = 7			
DM Index	3			
DM Val Before	0			
DM Val After	1475			

13 - and $s6, $s1, $s4

Inst.	opcode	$s1	$s4	$s6	shamt	funct
Code	000000	10001	10100	10110	00000	011011
Index	R-Type	17	20	22		And
Val Before		1382	1661	1847		
Val After		1382	1661	1124		

14- beq $zero, $a0, 0 True

Inst.	opcode	$zero	$a0	Offset
Code	100000	00000	00100	00000 00000 000000
Index	Beq	0	4	Next Inst.
Val Before		1	1	
Val After		1	1	

15- or $t7, $s5, $t4 Will BE Executed

Inst.	opcode	$s5	$t4	$t7	shamt	funct
Code	000000	10101	01100	01111	00000	110101
Index	R-Type	21	12	15		Or
Val Before		1754	921	1196		
Val After		1754	921	2011		

Data After Running Instructions:

Reg Name	Reg Index	Reg Value	DM Address	DM Index	DM Value
$zero	0	1	4	0	526
$at	1	526	5	1	1196
$v0	2	60	6	2	60
$v1	3	136	7	3	1475
$a0	4	1	8	4	0
$a1	5	0	9	5	0
$a2	6	381	10	6	0
$a3	7	3	11	7	0
$t0	8	559	12	8	0
$t1	9	648	13	9	0
$t2	10	739	14	10	0

$t3	11	2	15	11	0
$t4	12	921	16	12	0
$t5	13	1012	17	13	0
$t6	14	1103	18	14	0
$t7	15	2011	19	15	0
$s0	16	382			
$s1	17	1382			
$s2	18	1475			
$s3	19	1568			
$s4	20	1661			
$s5	21	1754			
$s6	22	1124			
$s7	23	1940			
$t8	24	2033			
$t9	25	2126			
$k0	26	2219			
$k1	27	2310			
$gp	28	2403			
$sp	29	2496			
$fp	30	1670			
$ra	31	2682			

3. Implementation

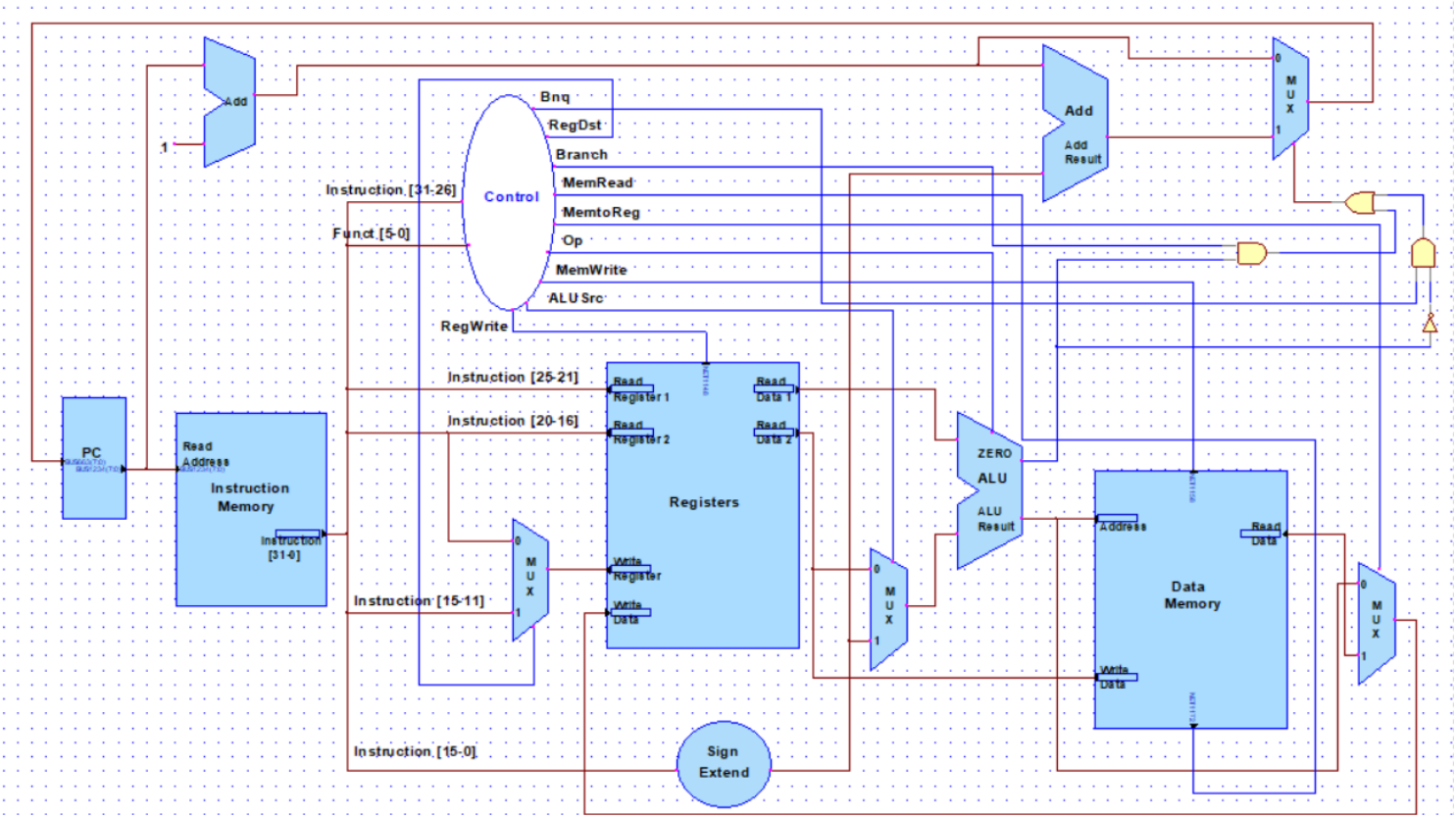
3.1 VHDL Code

Link to the GitHub repository MIPs

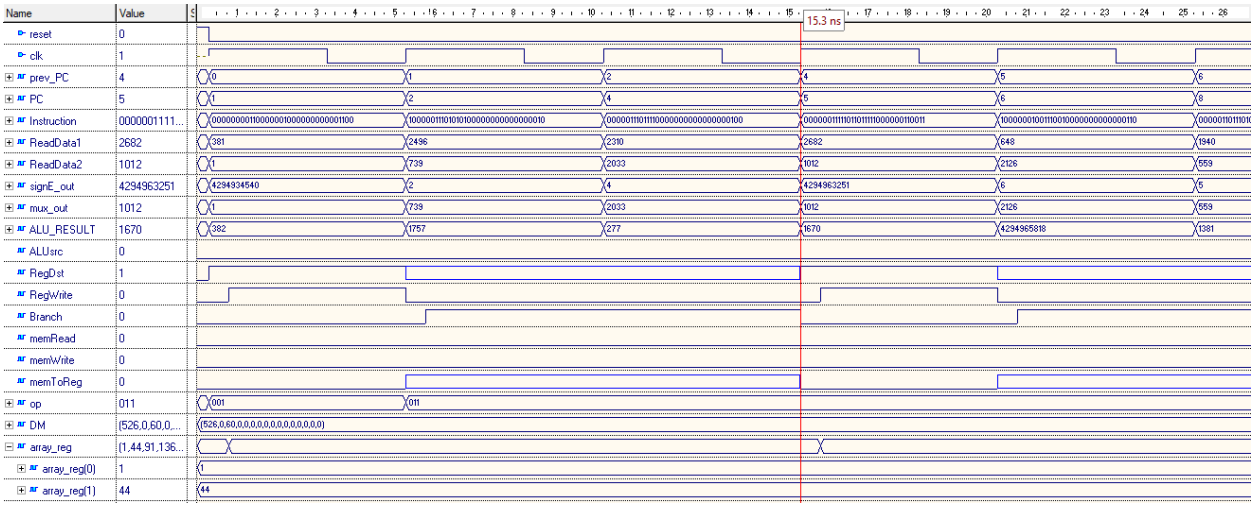
Or

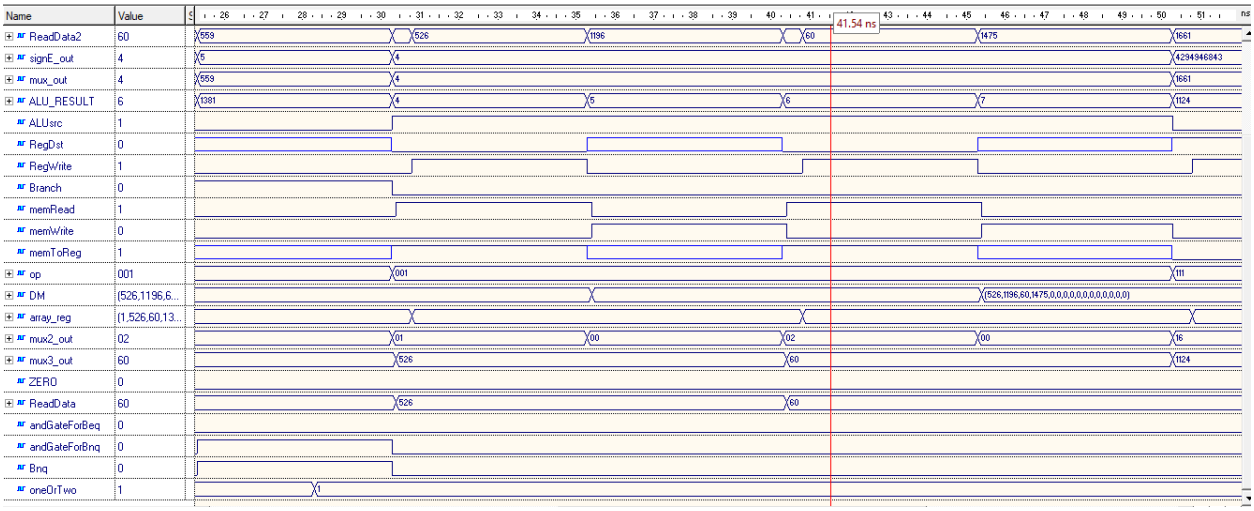
Scan to find the full code on GitHub!

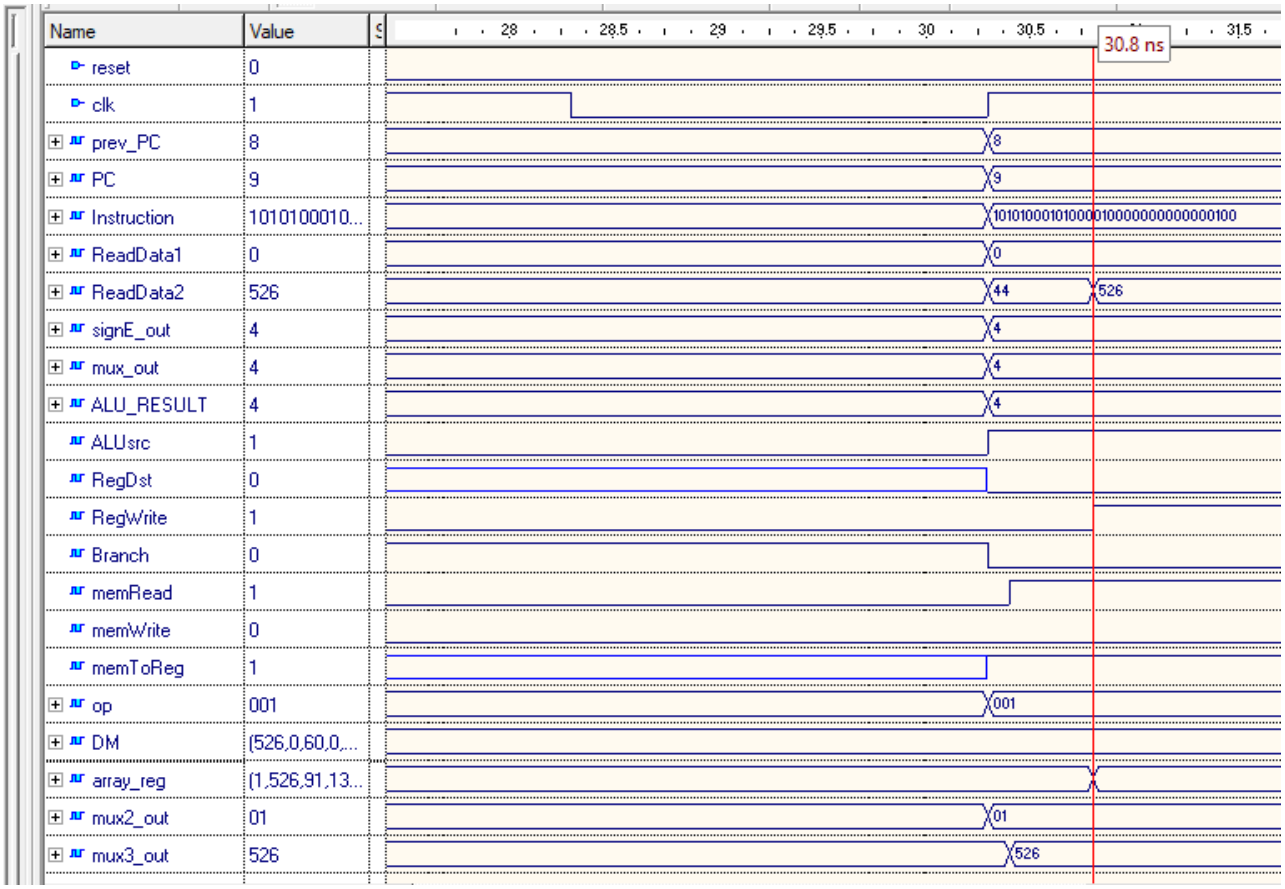
3.2 Give the Schematic diagram using Active HDL

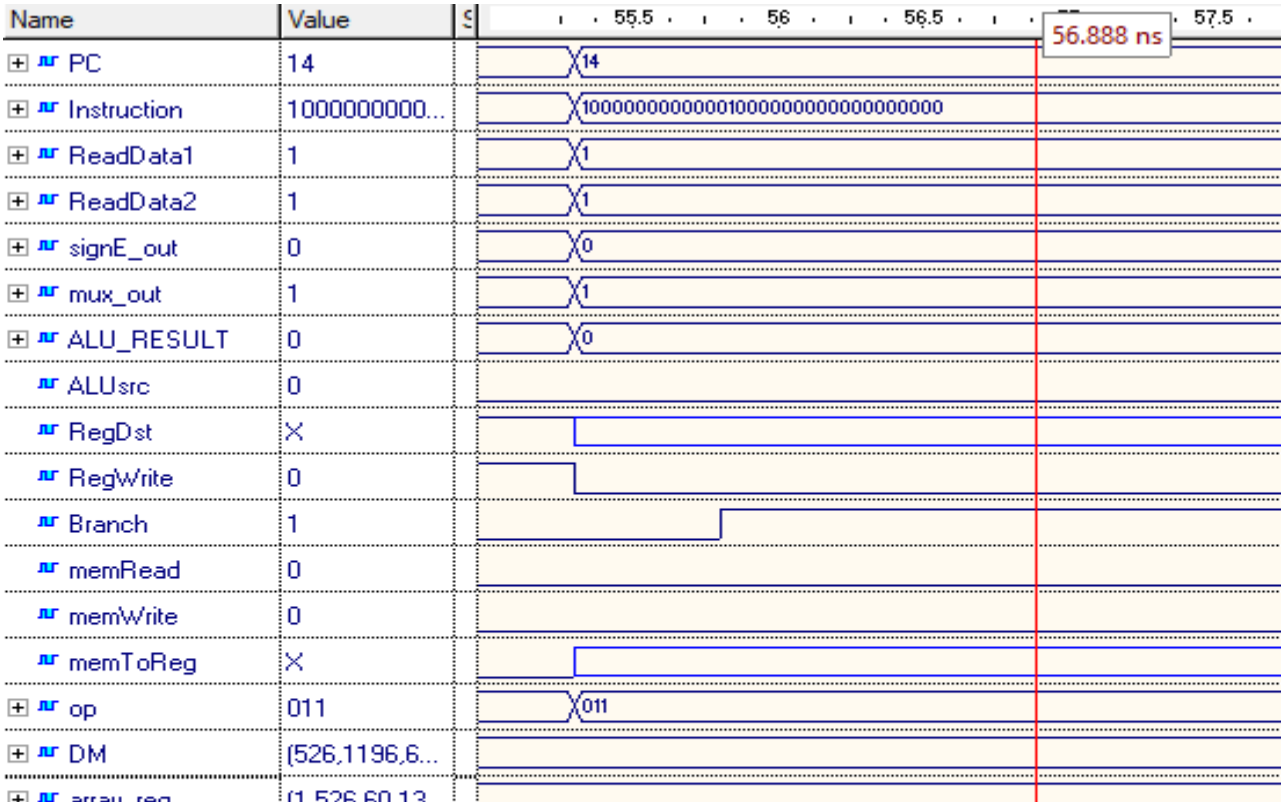


3.3 Simulate result






Name
Value
28
28.5
29
29.5
30
30.5
31.5
30.8 ns
reset
0
clk
1
prev_PC
8
PC
9
Instruction
1010100010...
ReadData1
0
ReadData2
526
signE_out
4
mux_out
4
ALU_RESULT
4
ALUsrc
1
RegDst
0
RegWrite
1
Branch
0
memRead
1
memWrite
0
memToReg
1
op
001
DM
(526,0,60,0,...
array_reg
(1,526,91,13...
mux2_out
01
mux3_out
526


Name
Value
55.5
56
56.5
57.5
56.888 ns
PC
14
Instruction
1000000000...
ReadData1
1
ReadData2
1
signE_out
0
mux_out
1
ALU_RESULT
0
ALUsrc
0
RegDst
X
RegWrite
0
Branch
1
memRead
0
memWrite
0
memToReg
X
op
011
DM
(526,1196,6...

3.3 References

Our main source:

- https://youtu.be/_z20VuxRzsc?si=8bzsJteZL7c0WV2B

For accurate information:

- https://drive.google.com/file/d/1tKK_pqbTbFHQJcMrKdjddlV0RxE1s9fu/view?usp=drivesdk
- https://drive.google.com/file/d/1tIBPstX9QccyiePPffKKtLBeDS_cNb3H/view?usp=drivesdk
- https://drive.google.com/file/d/1JbXpmKI3zJaiAnE-bP8GSXk9qai8jkAO/view?usp=sharing

THANK YOU