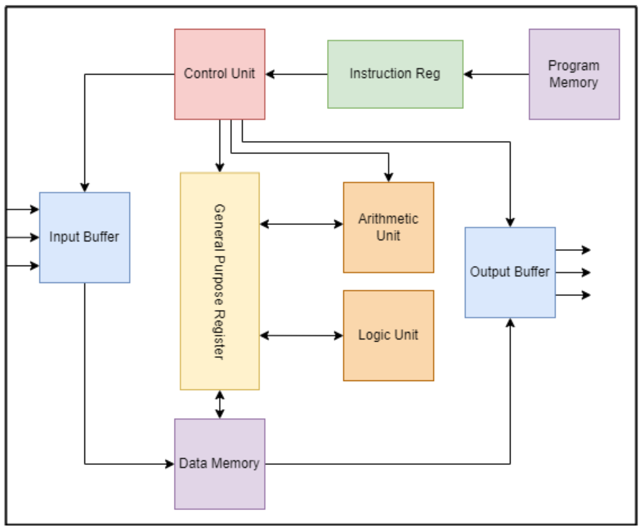
A 32-bit MIPS (Microprocessor without Interlocked Pipeline Stages) processor is a type of microprocessor architecture commonly used in embedded systems and early computer systems. MIPS processors are known for their simplicity, efficiency, and versatility.
Here is a description of the key design aspects of the implemented MIPS processor:

Architecture Overview:
- 32-bit RISC MIPS processor with 16-bit data path
- Harvard architecture with separate instruction and data memories



Registers:
- 32 general purpose 16-bit registers (GPR[31:0])
- 16-bit special purpose register (SGPR) to hold MSB of multiplication result
- 32-bit Instruction Register (IR)
- 16-bit Program Counter (PC)

Instruction Formats:
- R-type: Register-Register instructions
  - Format: [oper_type][rdst][rsrc1][imm_mode][rsrc2]
- I-type: Register-Immediate instructions
  - Format: [oper_type][rdst][rsrc1][imm_mode][isrc]

- 32-bit fixed length instructions
- Fields:
  - oper_type[31:27]: Operation type
  - rdst[26:22]: Destination register
  - rsrc1[21:17]: Source register 1
  - imm_mode[16]: Immediate mode select
  - rsrc2[15:11]/isrc[15:0]: Source register 2 or 16-bit immediate value

Instruction Set:
- Arithmetic (add, sub, mul)
- Logical (and, or, xor etc)
- Load/Store (load, store)
- Control (jump, branch)

Arithmetic	Op code
movsgpr	5'b00000
mov	5'b00001
add	5'b00010
sub	5'b00011
mul	5'b00100

Logical	Op code
or	5'b00101
and	5'b00110
xor	5'b00111
xnor	5'b01000
nand	5'b01001
nor	5'b01010
not	5'b01011

Load & Store	Op code
storereg	5'b01101
storedin	5'b01110
senddout	5'b01111
sendreg	5'b10001

Jump and Branch	Op code
jump	5'b10010
jcarry	5'b10011
jnocarry	5'b10100
jsign	5'b10101
jnosign	5'b10110
jzero	5'b10111
jnozero	5'b11000
joverflow	5'b11001
jnooverflow	5'b11010

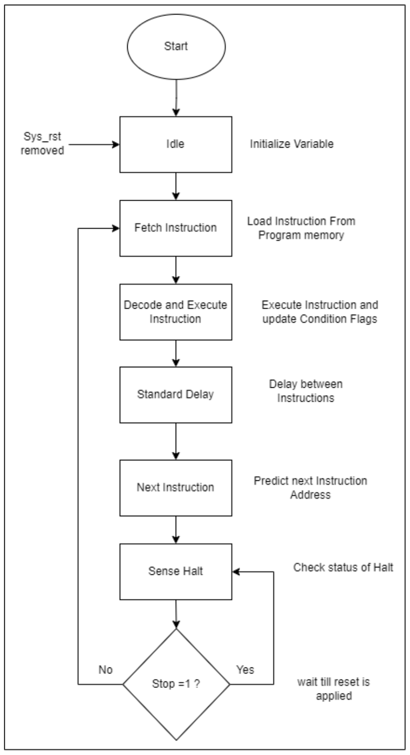
Halt	Op code
halt	5'b11011

Memory Hierarchy:
- Harvard architecture
- 16 x 32-bit Instruction Memory (inst_mem)
- 16 x 16-bit Data Memory (data_mem)
- No caching or memory hierarchy

Pipeline Stages:
The key pipeline stages are:

1. Fetch Instruction (fetch_inst)
2. Decode and Execute Instruction (dec_exec_inst)
3. Update PC and fetch next instruction (next_inst)

Control Unit:
- Finite state machine based control unit to transition between pipeline stages


Start
Sys_rst removed
Idle
Initialize Variable
Fetch Instruction
Load Instruction From Program memory
Decode and Execute Instruction
Execute Instruction and update Condition Flags
Standard Delay
Delay between Instructions
Next Instruction
Predict next Instruction Address
Sense Halt
Check status of Halt
No
Stop =1 ?
Yes
wait till reset is applied

FSM States:
- idle - Reset/idle state
- fetch_inst - Fetch instruction from memory
- dec_exec_inst - Decode and start executing instruction
- delay_next_inst - Delay state for execution completion
- next_inst - State for branch resolution and PC update for next instruction fetch
- sense_halt - Check for halt instruction

State Registers:
- state - Holds current FSM state

- next_state - Determines next state

Control Logic:
- next_state logic determines state transitions
- state register updates on clock edge
- Output logic generates control signals

Data Path:
- 16-bit data path
- Connects main components like registers, ALU, memories

Data Flow:
- Instruction fetched from inst_mem into IR
- GPR values moved to ALU input registers
- ALU output stored back into GPR
- Load/Store instructions transfer data between GPR and data_mem
- PC provides address into inst_mem for instruction fetch

Conditional Flags:
The following conditional flags have been implemented for the MIPS processor:

Carry Flag:
- Set when an arithmetic operation results in a carry out
- Used for add and subtract operations
- Set based on 17-bit temporary sum (temp_sum)

Sign Flag:
- Indicates sign of result
- Set to MSB of result register (GPR or SGPR)

Zero Flag:
- Indicates if result is zero
- Set by ANDing all bits of result register

Overflow Flag:
- Indicates arithmetic overflow
- Set based on operand and result MSBs
- Used for add and subtract instructions

The flags are set in the **decode_condflag task** based on the instruction type and result.

Jump and Branch
The key features of jump and branching in the MIPS processor code are:

Jump Instructions:

- `jump instruction for unconditional jump

- Jumps to 16-bit address specified in instruction
- Sets jmp_flag to take jump

Branch Instructions:

- Conditional branch instructions based on flag values
- E.g. jcarry, jsign, jzero, joverflow
- Compare flag value to determine if branch is taken
- Set jmp_flag if branch condition met

Branch Resolution:
- Branch target resolution in next_inst stage

PC Update:

- PC updated in next_inst stage
- PC incremented by 1 for sequential execution
- For jumps/branches, PC is set to target address

Load/Store:

Load Instructions:

1. sendreg: Loads data from data memory into a general-purpose register
2. senddout: Loads data from memory location into dout port

Store Instructions:

1. storereg: Stores register value into data memory
2. storedin: Stores immediate 16-bit value into memory

Branch Prediction: No branch prediction

Hazard Control: No hazard detection or control

Endianness: Little endian

Exception Handling: No exception handling logic