THE NATIONAL INSTITUTE OF ENGINEERING MYSORE

Autonomous Institue Affliated to VTU

Department of Electronics and communication Engineering

MINOR PROJECT PHASE -1 PRESENTATION ON

"Implementation of Y86-64 processor architetecture design using hardware coding language VERILOG "

Guided by :

Dr. Narasimha kaulgud

Professor

Department Of ECE

NIE MYSORE

Y86-64 ISA Processor

Project Description:

This project involves the implementation of Y-86 64 processor architecture design using hardware coding language VERILOG.

Steps involved in building the project:

Step-1: Sequential implementation.

Step-2: Pipeline implementation.

The aim of the project is to build a 5-stage pipeline implementation of Y86-64 processor with features like forwarding, halt, instructions.

Y86-64 ISA Processor

Introduction

The **Y86-64 ISA** is a simplified, educational version of the x86-64 instruction set architecture, designed to help students understand the core principles of computer architecture and instruction set design. Developed as a learning tool, Y86-64 provides a reduced set of instructions, addressing modes, and registers, making it easier to grasp the essential concepts behind modern processor architectures.

Key Features of Y86-64 ISA

Simplified Instruction Set

8-Byte Data Operations

General-Purpose Registers

Reduced Addressing Modes

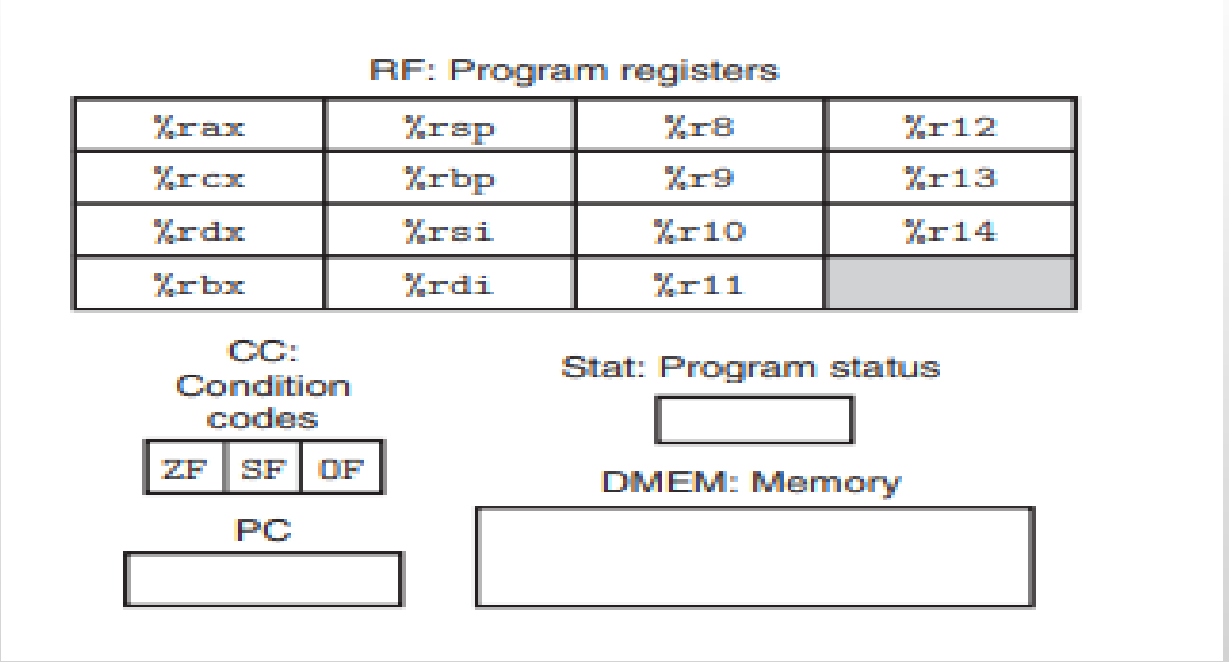
Instruction Encoding

Condition Codes and Control Flow

DEFINITION

Defining an instruction set architecture, such as Y86-64, includes defining the different components of its state, the set of instructions and their encodings, a set of programming conventions, and the handling of exceptional events.

1. Programmer-Visible State



2. Y86-64 Instructions

The set of Y86-64 instructions is largely a subset of the x86-64 instruction set. It includes only 8-byte integer operations, has fewer addressing modes, and includes a smaller set of operations. Since we only use 8-byte data, we can refer to these as “words” without any ambiguity

3. Instruction Encoding

Instruction encodings range between 1 and 10 bytes. An instruction consists of a 1-byte instruction specifier, possibly a 1-byte register specifier, and possibly an 8-byte constant word

Instruction	Byte offset from PC									
	0	1	2	3	4	5	6	7	8	9
halt	0 0									
nop	1 0									
cmovXX rA, rB	2 fn	rA rB								
irmovq V, rB	3 0	f rB	V							
rmmovq rA, D(rB)	4 0	rA rB	D							
mrmovq D(rB), rA	5 0	rA rB	D							

Instruction	Byte offset from PC								
	0	1	2	3	4	5	6	7	8
OPq rA, rB	6 fn	rA rB							
jXX Dest	7 fn	Dest							
call Dest	8 0	Dest							
ret	9 0								
pushq rA	a 0	rA f							
popq rA	b 0	rA f							

4. Y86-64 Exceptions

The programmer-visible state for Y86-64 includes a status code Stat describing the overall state of the executing program

Value	Name	Meaning
1	AOK	Normal operation
2	HLT	halt instruction encountered
3	ADR	Invalid address encountered
4	INS	Invalid instruction encountered

Sequential Y86-64 Implementations

As a first step, we describe a processor called SEQ (for “sequential” processor). On each clock cycle, SEQ performs all the steps required to process a complete instruction. This would require a very long cycle time, however, and so the clock rate would be unacceptably low

Stages in Y86-64

- Fetch
- Decode
- Execute
- Memory
- WriteBack
- PC update

Fetch.

The fetch stage reads the bytes of an instruction from memory, using the program counter (PC) as the memory address. From the instruction it extracts the two 4-bit portions of the instruction specifier byte, referred to as icode (the instruction code) and ifun (the instruction function). It possibly fetches a register specifier byte, giving one or both of the register operand specifiers rA and rB. It also possibly fetches an 8-byte constant word valC. It computes valP to be the address of the instruction following the current one in sequential order. That is, valP equals the value of the PC plus the length of the fetched instruction

Decode.

The decode stage reads up to two operands from the register file, giving values valA and/or valB. Typically, it reads the registers designated by instruction fieldsrA and rB, but for some instructions it reads register %rsp.

Execute

The core of instruction execution happens here. The ALU (Arithmetic Logic Unit) performs operations based on the instruction's function. For example, it could perform addition, subtraction, or logical operations on aluA and aluB to produce the result valE. Conditional codes (CC) determine whether conditions are met (for conditional instructions), generating a conditional signal (Cnd).

Memory.

The memory stage may write data to memory, or it may read data from memory. We refer to the value read as valM.

Write back.

The write-back stage writes up to two results to the register file.

PC update.

The PC is set to the address of the next instruction.

Why Y86-64?

Y86-64 is an ideal learning platform because it retains the essential structure of a modern ISA without the complexity of a full-scale x86-64 processor. Students can gain hands-on experience in implementing a basic CPU, learn about instruction encoding, and understand how higher-level programming constructs are translated into machine instructions. This simplified ISA is a stepping stone toward mastering more advanced computer architecture concepts and preparing for real-world processor design.

Limitation of Sequential Y86-64

1. Low Performance Due to Limited Instruction Throughput

•In a sequential implementation, each instruction is processed fully before the next one starts. This serial approach means only one instruction is completed at a time, resulting in low throughput.

•This bottleneck limits the processor's overall speed and efficiency, as no instructions overlap in their execution.

2.Poor Resource Utilization

•Sequential execution does not make efficient use of CPU resources (such as the arithmetic logic unit (ALU), memory, and registers).

•When the CPU is busy fetching or decoding an instruction, other parts of the processor, like the ALU, are idle, leading to under-utilization of resources and wasted cycles.

3. Increased Execution Time

•Each instruction must go through all the stages of execution (fetch, decode, execute, memory access, write-back) without overlap. This makes sequential execution slower than parallel or pipelined implementations.

•The increased instruction execution time can limit performance, especially in applications requiring high computational power.

4.Latency in Memory Access

•A sequential Y86-64 processor will encounter delays whenever there is a memory access operation, such as loading from or storing to memory.

•Since each instruction must wait until the memory access completes, the entire instruction sequence is delayed, further reducing efficiency.

References:

Computer Systems A Programmer's Perspective

Author :Randal E. Bryant (Carnegie Mellon University)

David R. O'Hallaron (Carnegie Mellon University)

Presented by

Muddu krishna Y 4NI22EC052

Mallappa Bagoji 4NI22EC046

Pannaga Kalkur 4NI22EC058

thank you