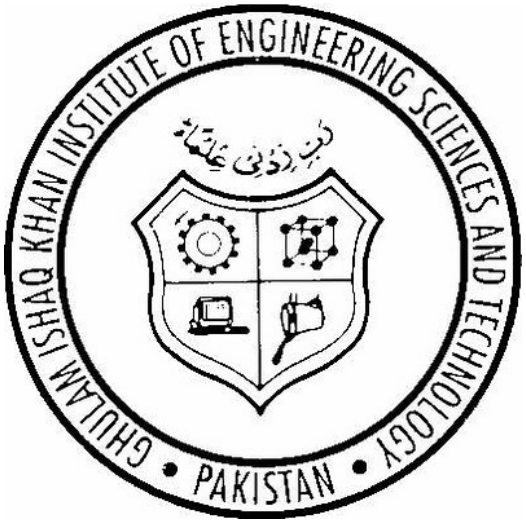

Computer Design with 17-bit Instruction Set

AUTHORS

Ayesha Kashif - 2022132
Mohammad Abdur Rehman - 2022299
Noor Ul Ain - 2022485

April 29, 2024

Contents

1 Abstract **1**

2 Introduction **2**

3 Instruction Set Architecture - ISA **3**
3.1 Arithmetic Instructions: . . . 3
3.2 Logic Instructions: . . . 3
3.3 Shift Instructions: . . . 3
3.4 I/O and Program Control Instructions: . . . 3
3.5 Instruction Format and Addressing Modes: . . . 4
3.6 Memory Size and Registers: . . . 4
3.7 Control of Registers, Flags, and Memory: . . . 5

4 Instruction Format and Addressing Mode **6**

5 Micro-Operations **7**
5.1 FETCH: . . . 7
5.2 DECODE: . . . 7
5.3 EXECUTE: . . . 7
5.3.1 Arithmetic Instructions: . . . 7
5.3.2 **Logic Instructions:** . . . 8
5.3.3 **Shift Instructions:** . . . 8
5.3.4 I/O and Program Control Instructions: . . . 9

6 Memory Size and Registers **10**
6.1 Memory Size: . . . 10
6.2 Registers: . . . 10

7 Control of Registers, Flags and memory **12**
7.1 Control Unit: . . . 12
7.2 Register Control: . . . 12
7.3 Flags: . . . 13
7.4 Memory Control: . . . 13
7.5 Control Signals: . . . 13

8 Bibliography **15**

1 Abstract

In the sphere of computer architecture, am exploring the creation of a hi-tech computer with the design that incorporates 17 bit instruction set. This system is our objective and it will be known for two major features: high efficiency and multifunctionality. It is going to be designed to be used in professional applications and in many different situations.

In our design the best option will be a clean, modestly time of the execution sets of instructions architecture which will be well thoughtfully made of many kind of arithmetic, logic, shift, input/output (I/O) and program flow functions. Which consequently promises broad-spectral computational task fulfilment, via maintaining a diligent structure, and optimizing for efficiency at the same time.

Key components of our computer design include:

- Implement a 17-bit instruction format, which will have certain of the optional addressing modes to facilitate data access and manipulation by having multiple ways of specifying the data source and destination.

- Moreover, our enhanced ALU (Arithmetic Logic Unit) is a brain behind the calculating device, designed to perform complex operations quickly and precisely.

- A set of registers, which are both general-purpose (thus, good for storing and manipulating data) and powerful (meaning they are suited for running programs quicker and with more versatility).

- What will be one of the most integral parts of this processor is that it will have a robust control unit for instruction execution, and reliable processor operation.

The other part of our computer design which is working in combination with the support for interrupt-driven and program-controlled I/O, is vital for the communicating with the external devices and peripherals. The processor contains an address memory capacity of 4096 and general-purpose 16 registers, so it provides a balance between storage and efficiency of computations, which expands its capabilities across many computing problems.

This covered the major points of my 17-bit instruction set design with its flexibility and its capability to exist in industrial computing environment.

2 Introduction

As for the computer architecture is concerned, designers usually seek a balanced approach, that is efficiency and simplicity, which happen most of the time by focusing on the essential operations. Our Essential Computer design on a 17-bit instruction set, the course and goal which are to create a process that helps in compaction while at the same time must be versatile enough to address basic computational tasks.

We base our design on an instruction set architecture (ISA) in which core arithmetic, logic, shift, I/O, and program control operations are found thus providing a robust platform for development. The main goal is to find an accurate compromise between functionalities and complexity in utmost sense of pressing just the right buttons, thus, presenting the refined set of instructions without sacrificing the computational power.

The instruction set of the processor contains math operations that are the building blocks of the numbers which are addition, subtraction, multiplication, and division. Moreover, the instructions set-complemented by logic operations such as AND, OR, XOR, and NOT. Shift commands yield useful tangential movement, leading to creating supplementary feedback of data.

Used to link without troubles with external devices gives input and output directions to the hat fastest command between the processor and I/O peripherals. If in case, opcode control instructions like jump, call, return, will be there, it will be convenient to let the instruction flow smoothly and guarantee proper subroutine handling.

To achieve these direction, we have developed an instruction format with opcode and operand fields, and these fields comprises of multiple addressing modes but also allows for flexible data and instruction accessibility.

The processor architecture includes the ALU for handling arithmetic & logic functions, some set of general-purpose registers, the dedicated control unit to assist in precision instructions execution management and the robust system bus to facilitate communication between memory, system devices as well as I/O. Our processor also includes flags helpful for displaying state information, for example, flag zero result or carry in operations of arithmetic.

The memory architecture and the usage of registers are the two essential points needed to be noted in case of our processor. Our processor has assumed a memory size of 4096 which translate into 16 general-purpose registers where the number of registers provides an optimal balance between storing data and processing it efficiently in the processor unit.

Moreover, our implementation is capable of independent common interruption and program-controlled I/O operations, which make it convenient for association with the external devices and retain the system responsiveness,

This summary sets up the presentation to an extent that thereafter follows a more detailed analysis of our Basic Computer design with the capability of 17-bit width instruction set, underscoring its superiority in precision, scalability, and professional setting of computing with low-power overhead.

3 Instruction Set Architecture - ISA

3.1 Arithmetic Instructions:

1. **ADD (Addition):** Opcode 0000 Performs addition of two operands.
2. **SUB (Subtraction)**:Opcode 0001 Performs subtraction of one operand from another.
3. **MUL (Multiplication)**:Opcode 0010 Multiplies two operands.
4. **DIV (Division):**Opcode 0011 Divides one operand by another.

3.2 Logic Instructions:

1. **AND (Logical AND)**: Opcode 0100 Performs bitwise AND operation between two operands.
2. **OR (Logical OR):** Opcode 0101 Performs bitwise OR operation between two operands.
3. **XOR (Logical XOR):** Opcode 0110 Performs bitwise XOR operation between two operands.
4. **NOT (Logical NOT):** Opcode 0111 Performs bitwise NOT operation on an operand.

3.3 Shift Instructions:

1. **LSHIFT (Logical Left Shift):** Opcode 1000 Shifts the bits of an operand to the left by a specified number of positions.
2. **RSHIFT (Logical Right Shift):** Opcode 1001 Shifts the bits of an operand to the right by a specified number of positions.

3.4 I/O and Program Control Instructions:

1. **IN (Input):** Opcode 1010 Reads data from an input device into a register.
2. **OUT (Output):** Opcode 1011 Writes data from a register to an output device.
3. **JMP (Jump):** Opcode 1100 Unconditionally jumps to a specified memory address.
4. **JZ (Jump if Zero):** Opcode 1101 Jumps to a specified memory address if the zero flag is set.
5. **JC (Jump if Carry):** Opcode 1110 Jumps to a specified memory address if the carry flag is set.
6. **CALL (Call Subroutine):** Opcode 1111 0000 Calls a subroutine at a specified memory address.

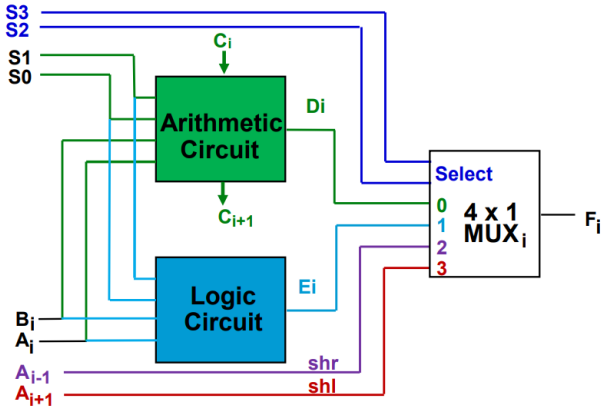

Figure 1: ALU

7. **RET (Return from Subroutine):** Opcode 1111 0001 Returns from a subroutine.

Basic structure of Arithmetic and Logic Unit (ALU) is shown in fig. 1

3.5 Instruction Format and Addressing Modes:

1. **Fixed-Length Format:** 17 bits Opcode (4 bits) Operand (12 bits) Direct/Indirect (1 bit)
2. **Addressing Modes:**

- **Register Addressing:** Accesses data directly from registers.
- **Immediate Addressing:** Uses immediate values as operands.
- **Direct Addressing:** Accesses data directly from memory addresses.

3.6 Memory Size and Registers:

1. **Memory Size:** 4096 (12-bit addressing)
2. **Registers:** 16 general-purpose registers (4-bit addressing)

3.7 Control of Registers, Flags, and Memory:

1. **Control Unit:** Manages instruction execution, ALU operations, register transfers, and memory operations.
2. **Flags:** Zero flag, carry flag, etc., set based on operation results.
3. **Interrupts and I/O:** Supports interrupt-driven and program-controlled I/O operations for efficient external device interaction.

4 Instruction Format and Addressing Mode

Instruction Format: Our simplified processor adopts a fixed-length instruction format to streamline instruction decoding and execution. Each instruction is 17 bits long, structured into two main fields and 1 bit is reserved for specifying addressing mode:

1. **Opcode Field (4 bits):** Specifies the operation to be performed by the processor. This field uniquely identifies each instruction and guides the control unit in initiating the corresponding microoperations.

2. **Operand Field (12 bits):** Contains the operand or data required for the instruction's execution. Depending on the instruction type, the operand field may represent a register address, an immediate value, or a memory address.

Addressing Modes: Our processor supports multiple addressing modes to enhance flexibility in accessing data and instructions from various sources. The addressing modes include:

1. **Register Addressing:**

   - **Description:** In this mode, the operand field of the instruction directly specifies one of the 16 general-purpose registers available in the processor.
   - **Example:** `ADD R1, R2` adds the contents of register R2 to register R1.

2. **Immediate Addressing:**

   - **Description:** Immediate addressing involves using a constant value embedded within the instruction as the operand.
   - **Example:** `ADD R1, #10` adds the value 10 to the contents of register R1.

3. **Direct Addressing:**

   - **Description:** Direct addressing allows the processor to access data directly from memory using the memory address provided in the operand field.
   - **Example:** `LOAD R1, 0x2000` loads the data from memory address 0x2000 into register R1.

By having various addressing methods, it gives the peripheral a diverse range of capabilities in terms of data handling and program management by having the efficiency needed for these tasks to be done. Because the automation of tasks can be better managed in terms of execution, with fewer resources required. The fixed format instruction length along with the device operand addressing modes present effective combination of instruction. first, there is energy conservation, secondly, they help in an efficient allocation of space and finally, they allow for increased efficiency in the process.

5 Micro-Operations

Fetch and decode instruction for each operation is same and listed below:

5.1 FETCH:

T0: AR $\leftarrow PC$
T1: IR $\leftarrow M[AR], PC \leftarrow PC + 1$

5.2 DECODE:

T2: D0, . . . , D7 $\leftarrow DecodeIR(12-15), AR \leftarrow IR(0-11)I \leftarrow IR(16)$
Direct address T3: Nothing
Indirect address AR $\leftarrow M[AR]$

5.3 EXECUTE:

5.3.1 Arithmetic Instructions:

1. ADD
D0T4: DR $\leftarrow M[AR]$
D0T5: AC $\leftarrow AC + DR, SC \leftarrow 0, E \leftarrow Cout$

2. SUB
D1T4: DR $\leftarrow M[AR]$
D1T5: AC $\leftarrow DR$
D1T6: AC $\leftarrow AC, AC \leftarrow AC + 1$
D1T7: AC $\leftarrow AC + DR, SC \leftarrow 0, E \leftarrow Cout$

3. MUL
D2T4: AC $\leftarrow 0$
D2T5: AC $\leftarrow M[R1]$
D2T6: if (R2 == 0) then (R3 $\leftarrow 0, PC \leftarrow PC + 1, SC \leftarrow 0$) *else go to D2T7*
D2T7: AC $\leftarrow AC + R2$
D2T8: R3 $\leftarrow AC$
D2T9: R2 $\leftarrow R21$
D2T10: if (R2 = 0) then (P C $\leftarrow PC + 1), SC \leftarrow 0$ *else go to D2T7*

4. DIV
D3T4: AC $\leftarrow R1$
D3T5: R3 $\leftarrow 0$
D3T6: AC $\leftarrow AC - R2$
D3T7: if (AC >= 0) then (R3 $\leftarrow R3 + 1, PC \leftarrow PC + 1), SC \leftarrow 0$ *else go to D3T6*
D3T8: Halt or PC $\leftarrow PC + 1, SC \leftarrow 0$

5.3.2 Logic Instructions:

1. AND
D4T4: DR $\leftarrow M[AR]$
D4T5: AC $\leftarrow AC \wedge DR, SC \leftarrow 0$
2. OR
D5T4: DR $\leftarrow M[AR]$
D5T5: AC $\leftarrow AC \vee DR, SC \leftarrow 0$

3. XOR
D6T4: DR1 $\leftarrow M[AR1]$
D6T5: DR2 $\leftarrow M[AR2]$
D6T6: DR3 $\leftarrow DR1 \wedge (DR2')$
D6T7: DR4 $\leftarrow (DR1') \wedge DR2$
D6T8: DR5 $\leftarrow DR3 \vee DR4$
D6T9: M[AR1] $\leftarrow DR5, SC \leftarrow 0$

4. NOT
D7T4: DR $\leftarrow M[AR]$
D7T5: DR $\leftarrow DR, M[AR] \leftarrow DR, SC \leftarrow 0$

5.3.3 Shift Instructions:

1. SHR
rB1: AC $\leftarrow shlAC, AC(0) \leftarrow E, E \leftarrow AC(16)$

2. SHL
rB2: AC $\leftarrow shrAC, AC(16) \leftarrow E, E \leftarrow AC(0)$

5.3.4 I/O and Program Control Instructions:

1. INP
pB10: AC(0 â 7) $\leftarrow INPR, FGI \leftarrow 0$

2. OUT
pB11: OUT R $\leftarrow AC(07), FGO \leftarrow 0$

3. JMP
D12T4: PC $\leftarrow AR, SC \leftarrow 0$

4. JEZ
D13T4: DR $\leftarrow M[AR]$
D13T5: M[AR] $\leftarrow DR, if(DR = 0)then(PC \leftarrow PC + 1), SC \leftarrow 0$

5. JC
D14T4: (if Cout = 1) then P C $\leftarrow AR, SC \leftarrow 0$

6 Memory Size and Registers

6.1 Memory Size:

The memory space of 4096 addresses is logically and physically divided into parts that are made accessible by the 17-bit instruction set of the Computer our design team has based its efforts on. It is composed of special exclusively for running instructions, programs, and subroutine the calls and storage, all geared towards speeding up both computing and memory operations.

The 4096 memory area is characterized by a big capacity that enables many alternatives to run their applications with various complexities without overwhelming the processor functionality. The addressing scheme in memory is defined with accurate precision, with a 17-bit word size allowing a maximum of 4096 total memory locations. Each cell is consist of a 17 bits word that means the address size is comparable to the word's size which gives us such a lot of opportunities to store both instructions and data and in this way make our processor easier and more efficient.

We follow the addressed way of structure dedicated to the linear memory model, where each location in the memory has its own 17-bit addresses starting from 0x0000 ending at 0xFFFF. It is this direct structure which makes memory management and access control more practical, and hence effective, within the processor architecture, consequently facilitating both developers and system administrators' task.

This fine level of detail in memory arrangement shows efficiency and precision of our Boilerplate Computer with a 17-bit instruction set once again stressing the coordination capabilities of the design and smoothness if its operation all over the running computing tasks and programs.

6.2 Registers:

Our processor architecture is meticulously designed to incorporate 16 highly capable general-purpose registers, each capable of holding a 17-bit data word. These registers serve as the backbone of efficient data manipulation, arithmetic operations, and program control, playing a pivotal role in optimizing computational performance.

Designated as R0 through R15, these general-purpose registers offer a total of 16 storage locations that are directly accessible by the processor's instructions. They act as efficient temporary storage for operands, intermediate computation results, and memory addresses during program execution, enhancing overall efficiency and speed.

One of the standout features of our design is the register addressing mode, which enables instructions to operate directly on data stored within registers. This approach significantly minimizes memory access times, thereby boosting computational efficiency and speeding up program execution. Additionally, the availability of multiple registers facilitates parallel processing of data, enabling our processor to efficiently handle complex algorithms and computations with ease.

By combining an optimal memory size of 4096 with 16 high-performance general-purpose registers, our processor design strikes a perfect balance between memory capacity and register utiliza-

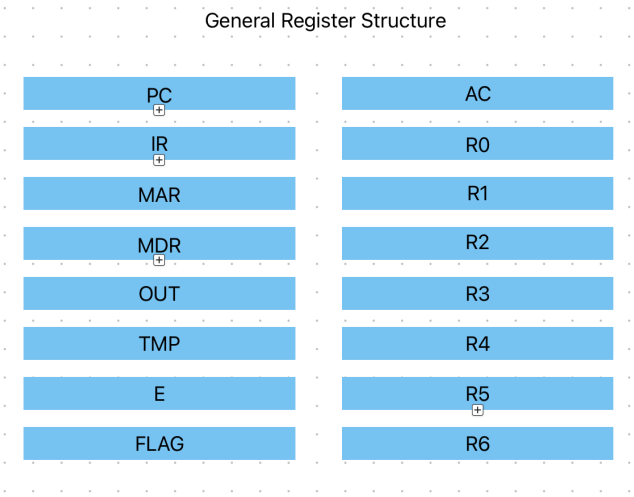

Figure 2: Register structure

tion. This equilibrium ensures not only efficient program execution but also remarkable versatility in handling a wide variety of computational tasks, making our processor design a standout choice for professional computing environments.

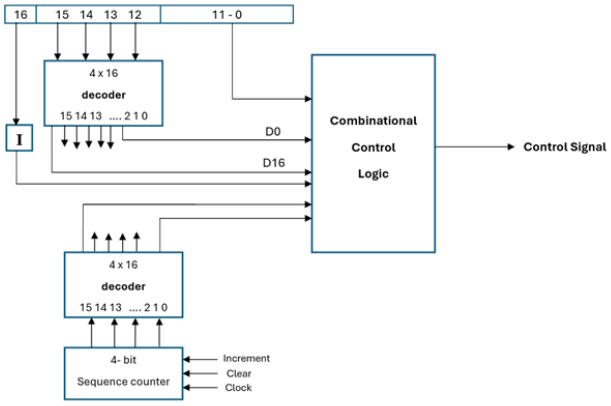

Figure 3: Control Unit

7 Control of Registers, Flags and memory

7.1 Control Unit:

The key structuring element is the control unit in the minimalistic edifice of simple Computer Design that acts as a fundamental link that is responsible for the coordination of instructions execution, managing data transition between registers and memory, and overseeing other internal operations. Control Unit acts as the main directing mechanism of the processor, where it performs the responsibility of interpretation of the codes retrieved from the memory, implementing various control signals, and supervising the complex set of microoperation executions. This key element works within the setting of a 17- bit instruction set which is dedicated to ensure better efficiency in instruction execution and so increases the system performance. The vital role of the Control Unit is in the proper interpretation and execution of data because by this, it'significantly contributes to the smooth and harmonious coordination within the processor between its different components, which is inclusive data transfers and operational continuity.

To sum up, the Control Unit is the micromanager that orchestrates the activities of our simple Computer Design perfectly such that the desired instructions are executed accurately and relevant data is managed and that internal communication takes place conveniently to bring about a perfect performance.

Basic structure of Control Unit is shown in fig. 3

7.2 Register Control:

Conversely, the Control Unit, as the higher commander within the hierarchy of the CPU, controls the transition of data between registers and execution units. It performs actions such as instantiation of data from registers, storing results, and others stem from registers. Therefore, efficient encoding

and decoding of data, proving storage space for larger programs, and managing register states during program execution. Computerized registration and itâs management guarantees optimal usability.

7.3 Flags:

Our processor has a wide spectrum of flags to send messages or enable decisions. Conditional branching is one of the features it has.Common flags include:

- **Zero Flag (Z):** Indicates if the result of an operation is zero, thus, resets the internal CPU registers to zero (reset to zero) and places zeros into the accumulator, memory, and registers.
- **Carry Flag (C):** Carrys out arithmetic operations and tells whether a carry or borrow happened in the process.
- **Overflow Flag (V):** This signals signed arithmetics overflow.
- **Sign Flag (S):** Besides, these two topics discuss the sign for the result.
- **Parity Flag (P):** Displays a minus sign to indicate the parity (even or odd) of the outcome. Create your own unique and state-of-the-art website. Instantly create and download text, audio, and video files.

The Control Unit establishes and, from time to time, updates the flags, which imply the outcomes of arithmetic, logic, and comparison computations. Flags functionify the conditional questio branching instructions that allow the process to depend on the outcome of precept operations.

7.4 Memory Control:

The Control Unit in the Basic Computer Design receives input instructions from the program and carries them out by utilizing fast electronic switching circuits for correctly managing memory operations that require reading and writing data from and to memory. It performs these acts by managing memory addresses, control the process of data transfers between the core and the memory, and ensuring the data quality during the access to the memory. Together, these operations pave the way for the stable and seamless operation of the system, getting rid of all operational problems and enabling the system function as desired.

7.5 Control Signals:

For execution of commands properly, Control Unit has to form control signals that causes particular device to be activated within the CPU. The data flow control schemes in the CPU are micro signals represented by a clock for synchronization timing, enable signals for activating the functional units (such as the ALU and the memory interface), and control signals for data multiplexing and routing.

In conclusion, the Control Unit is the central entity that carries the critical responsibility of ensuring adequate coordination of registers, flags to monitor the status of the program and instructions on accessing the memory to successfully complete tasks in our designed reduced instruction set computing processor architecture.

8 Bibliography

1. Hennessy, J. L., & Patterson, D. A. (2011). Computer Architecture: A Quantitative Approach (5th ed.). Morgan Kaufmann. This book provides a comprehensive understanding of computer architecture principles, including RISC architecture concepts, instruction set design, and memory management.

2. Tanenbaum, A. S., & Goodman, J. (2002). Modern Operating Systems (3rd ed.). Prentice Hall. The chapter on computer architecture in this book offers insights into control unit design, memory management, and processor control mechanisms.

3. Hamacher, V. C., Vranesic, Z. G., & Zaky, S. A. (2001). Computer Organization and Embedded Systems (5th ed.). McGraw-Hill. This textbook covers topics related to processor design, instruction formats, addressing modes, and control unit operations.

4. Stallings, W. (2017). Computer Organization and Architecture: Designing for Performance (10th ed.). Pearson. The chapters on RISC architectures, memory systems, and control units provide valuable information for understanding processor design principles.

5. Patterson, D. A., & Waterman, A. (2018). The RISC-V Reader: An Open Architecture Atlas. Morgan & Claypool Publishers. This book offers insights into the RISC-V instruction set architecture, which can be helpful in designing a simplified RISC processor.

6. Patterson, D. A., & Hennessy, J. L. (2017). Computer Organization and Design: The Hardware/Software Interface (5th ed.). Morgan Kaufmann. The book covers fundamental concepts of computer organization, including instruction set design, memory hierarchy, and control unit operation.