MIPS

Table of contents

INTRODUCTION TO MIPS:

MIPS (Microprocessor without Interlocked Pipeline Stages) stands as a cornerstone in the realm of computing, renowned for its efficiency and versatility. Originating in the 1980s, MIPS architecture has left an indelible mark on various computing devices, from personal computers to embedded systems. This introduction aims to provide a succinct overview of MIPS architecture, shedding light on its fundamental principles and significance in the computing landscape.

1. **Origins and Evolution**: Delve into the inception of MIPS architecture, tracing its roots back to Stanford University. Explore its evolution over the years, highlighting key milestones and innovations that have shaped its trajectory.

2. **Core Principles**: Unravel the core principles underpinning MIPS architecture, including Reduced Instruction Set Computing (RISC) philosophy, which emphasizes simplicity and efficiency in instruction execution.

3. **Key Components**: Introduce the essential components of a MIPS processor, such as the instruction set architecture (ISA), registers, memory organization, and pipeline stages. Illustrate these components with clear diagrams to aid comprehension.

4. **Performance and Efficiency**: Discuss how MIPS architecture achieves superior performance and efficiency through streamlined instruction execution, reduced complexity, and optimized pipeline design.

5. **Application Domains**: Explore the diverse application domains where MIPS architecture finds prominence, from consumer electronics and networking devices to automotive systems and beyond.

6. **Impact and Legacy**: Reflect on the enduring impact and legacy of MIPS architecture, recognizing its influence on subsequent processor designs and its continued relevance in modern computing.

DECODING MIPS ARCHITECTURE:

MIPS instruction format and syntax are the bedrock of programming for this architecture. Let's delve into this integral aspect with clarity and brevity:

1. **Instruction Format Overview**: MIPS instructions typically follow a fixed format, comprising fields for opcode, source and destination registers, immediate values, and memory addresses.

2. **Opcode**: The opcode specifies the operation to be performed, such as arithmetic, logical, or data transfer.

3. **Registers**: MIPS architecture boasts a plethora of general-purpose registers denoted by $s, $t, and $d, facilitating efficient data manipulation and storage.

4. **Immediate Values**: Immediate values allow for operations with constant data, providing flexibility in arithmetic and logical operations.

5. **Memory Addresses**: Instructions involving memory access utilize memory addresses to read from or write to specific locations in the memory hierarchy.

6. **Syntax Clarity**: MIPS assembly language adheres to a clear and consistent syntax, making it easy to understand and write code.

- **INSTRUCTION FORMAT AND SYNTAX**

1. Arithmetic and Logical Operations:

mnemonic	number of operands	operands	C or C++ or Java
move	2	d, s1	d = s1;
add	3	d, s1, s2	d = s1 + s2; two's complement
addu	3	d, s1, s2	d = s1 + s2; unsigned
sub	3	d, s1, s2	d = s1 - s2; two's complement
subu	3	d, s1, s2	d = s1 - s2; unsigned
mul	3	d, s1, s2	d = s1 * s2; two's complement
div	3	d, s1, s2	d = s1 / s2; gives quotient
divu	3	d, s1, s2	d = s1 / s2; gives quotient
rem	3	d, s1, s2	d = s1 % s2; gives remainder
remu	3	d, s1, s2	d = s1 % s2; gives remainder
and	3	d, s1, s2	d = s1 & s2; bitwise AND
or	3	d, s1, s2	d = s1 | s2; bitwise OR
not	2	d, s1	d = ~s1; bitwise complement
nand	3	d, s1, s2	d = s1 NAND s2; no C equivalent
nor	3	d, s1, s2	d = s1 NOR s2; no C equivalent
xor	3	d, s1, s2	d = s1 ^ s2; bitwise XOR
rol	3	d, s1, s2	d = rotate left of s1 by s2 places
ror	3	d, s1, s2	d = rotate right of s1 by s2 places
sll	3	d, s1, s2	d = logical left shift of s1 by s2 places
sra	3	d, s1, s2	d = arithmetic right shift of s1 by s2 places
srl	3	d, s1, s2	d = logical right shift of s1 by s2 places

Examples:

```
move  $4, $9          # copy contents of $9 into $4

mul   $12, $13, $14 # place 32-bit product of $13 and $14 into $12
                      # does not work correctly if result requires
                      # more than 32 bits

add   $8, $9, $10     # two's complement sum of $9 and $10 placed in $8

add   $20, $20, 1     # add (immediate value) 1 to the value in $20,
                      # result goes to $20
```

2.Control Instructions

Control instructions are the branches and/or jumps.

```
beq       3       r1, r2, label   branch to label if (r1) == (r2)
bne       3       r1, r2, label   branch to label if (r1) != (r2)
bgt       3       r1, r2, label   branch to label if (r1) > (r2)
bge       3       r1, r2, label   branch to label if (r1) >= (r2)
blt       3       r1, r2, label   branch to label if (r1) < (r2)
ble       3       r1, r2, label   branch to label if (r1) <= (r2)
```

PROGRAMMING SAMPLE :

```
add $t0, $r2, $r3
sub $t1, $r5, $r6
div $r4, $r5
mflo $t2
add $r7, $t0, $t1
add $r7, $r7, $t2
```

- $t0 = $r2 + $r3: Adds the values stored in registers $r2 and $r3 and stores the result in register $t0.
- $t1 = $r5 - $r6: Subtracts the value in register $r6 from the value in register $r5 and stores the result in register $t1.
- Divide $r4 by $r5: Divides the value in register $r4 by the value in register $r5.
- $t2 = Quotient of $r4 / $r5: Stores the quotient of the division operation performed in step 3 in register $t2 using the mflo instruction.
- $r7 = ($t0 + $t1): Adds the values stored in registers $t0 and $t1 and stores the result in register $r7.
- $r7 = ($r7 + $t2): Adds the value stored in register $t2 to the value already stored in register $r7 and stores the final result in register $r7.

Recent Trends in MIPS Technology

In the dynamic realm of technology, MIPS architecture continues to evolve, adapting to meet the demands of modern computing. Here, we highlight the recent trends shaping the trajectory of MIPS technology in a clear and concise manner.

1. **Enhanced Performance**: Recent advancements in MIPS architecture focus on boosting performance through optimizations in instruction execution, memory access, and pipeline efficiency.

2. **Energy Efficiency**: With an increasing emphasis on energy conservation, recent developments in MIPS technology prioritize energy-efficient designs, catering to the growing demand for sustainable computing solutions.

3. **Integration of AI**: As artificial intelligence (AI) becomes ubiquitous, MIPS architecture is incorporating AI accelerators and specialized instructions to support AI workloads efficiently, enabling a new era of intelligent computing.

4. **Security Enhancements**: With cybersecurity threats on the rise, MIPS processors are integrating enhanced security features such as hardware-based encryption, secure boot mechanisms, and robust memory protection schemes to safeguard sensitive data and ensure system integrity.

5. **Embedded Systems and IoT**: MIPS architecture continues to dominate the realm of embedded systems and Internet of Things (IoT) devices, with recent trends focusing on enhancing connectivity, reducing power consumption, and enabling seamless integration with IoT ecosystems.

CONCLUSION:

In conclusion, MIPS architecture represents a powerful blend of efficiency, versatility, and scalability in the computing landscape. Its streamlined design, rooted in the principles of Reduced Instruction Set Computing (RISC), enables high-performance computing across a diverse

range of applications. While it faces challenges in market share and mobile adoption, MIPS continues to evolve, embracing trends such as energy efficiency, AI integration, and open-source collaboration. As technology advances, MIPS architecture remains poised to play a significant role in shaping the future of computing, offering a robust foundation for innovation and progress.

Pepared by M.M.Sampath Ram