CSE301 – Computer Organization

Lecture 3 Notes

MIPS Memory Addressing and Alignment

- **Memory model:** Single-dimensional array
- **Addressing:** Byte-addressable (each byte has its own address)
- **Data sizes:**
  - Word = 32 bits = 4 bytes
  - Halfword = 16 bits = 2 bytes
- **Byte order:** Big-endian (MSB stored at lowest memory address)
- **Alignment rule:**
  - Word addresses must be **multiples of 4**
  - Halfword addresses must be **multiples of 2**

> **Check your Understanding**
>
> Is it legal to load a **word** from these addresses?
>
> - `0x0000_FFF4`
> - `0xFFFF_EE22`
> - `0xFF00_000F`

MIPS Registers

- **Total registers:** 32 general-purpose registers, each **32 bits** wide
- **Register addressing:** Requires **5 bits** to identify one of 32 registers
- **Performance tip:** Keep variables in registers whenever possible for faster access
- **Spilling registers:** When registers are full, less frequently used variables are temporarily stored in **memory** — this process is called **register spilling**

Instructions

Instruction Categories

MIPS instructions are grouped into five main types:

- **Arithmetic** — Perform mathematical operations
- **Data Transfer (Load/Store)** — Move data between memory and registers
- **Logical** — Perform bitwise operations
- **Conditional Branch** — Change control flow based on condition
- **Unconditional Jump** — Jump directly to a target address

Instruction Formats

All MIPS instructions are **exactly 32 bits wide** and follow one of three formats:

Format	Name	Used For
R-Type	Register	Arithmetic and logical operations between registers
I-Type	Immediate	ALU operations with constants, load/store, conditional branches

J-Type	Jump	Unconditional jumps

R-Type Format

- **Opcode:** 000000 for all R-type instructions
- **Used for:** Register-to-register ALU operations

Fields:

Field	Meaning
rs	Source register 1
rt	Source register 2
rd	Destination register
shamt	Shift amount (for shift instructions)
funct	Function code (specifies exact operation)

I-Type Format

- Contains a **16-bit immediate constant** inside the instruction
- Used for **ALU operations with immediates**, **load/store**, and **conditional branches**

Fields (by usage):

Type	rs	rt	Immediate/Offset
ALU Immediate	Source operand	Destination register	Constant operand
Load/Store	Base address register	Destination (load) or source (store)	16-bit byte offset
Branch	First register for comparison	Second register	16-bit word offset added to PC

J-Type Format

- **Fields:**
  - op: Opcode (specifies jump instruction type)
  - target: Immediate constant giving jump **target address (in words)**
- **Used for:** Unconditional jumps (j, jal)

MIPS Addressing Modes

1. **Immediate Addressing**
   - Format: op rs rt immediate
   - Example: addi $1, $2, 100
   - The operand is a constant value (immediate) encoded directly in the instruction.

2. **Register Addressing**

    - Format: op rs rt rd ... funct
    - Example: add $1, $2, $3
    - Both operands are in registers.

3. **Base (Displacement) Addressing**

    - Format: op rs rt address
    - Example: lw $1, 32($2)
    - The effective memory address = base register ($2) + offset (32).

4. **PC-relative Addressing**

    - Format: op rs rt address
    - Example: beq $1, $2, 100
    - The effective address = PC + offset × 4.
    - Used mainly for branch instructions.

5. **Pseudodirect Addressing**

    - Format: op address
    - Example: j 10000
    - The target address is formed by combining part of the current PC with the address field in the instruction.