UCCD1133
Introduction to Computer Organisation and Architecture

Assignment Project Exam Help

https://powcoder.com

Chapter 4
Computer Architecture and Organisation Fundamental

Add WeChat powcoder

Slides adapted from http://booksite.elsevier.com

Disclaimer

- This slide may contain copyrighted material of which has not been specifically authorized by the copyright owner. The use of copyrighted materials are solely for educational purpose. If you wish to use this copyrighted material for other purposes, you must first obtain permission from the original copyright owner.

Assignment Project Exam Help

https://powcoder.com

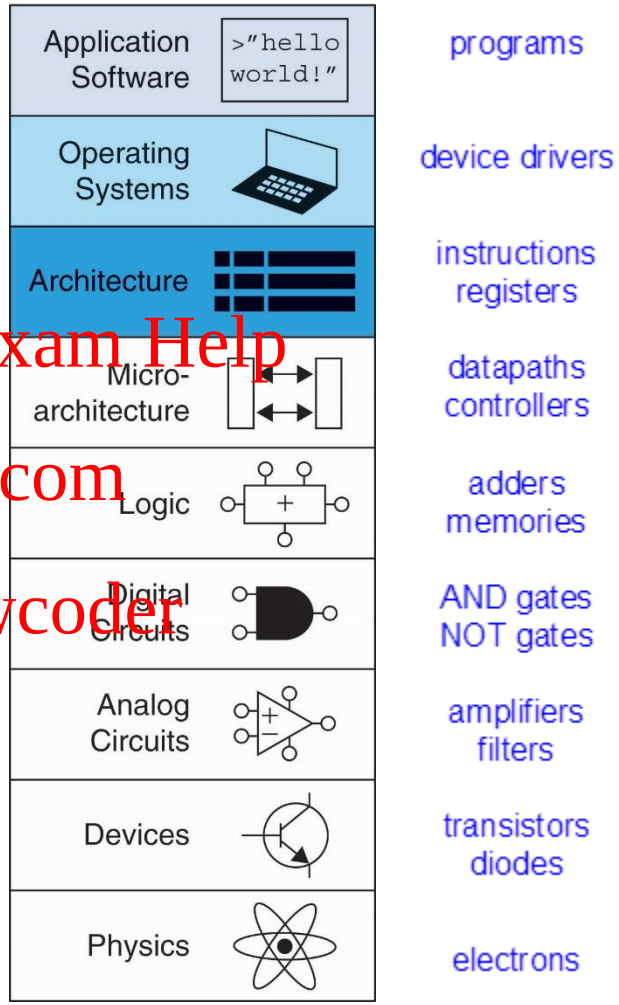
Add WeChat powcoder

Outline

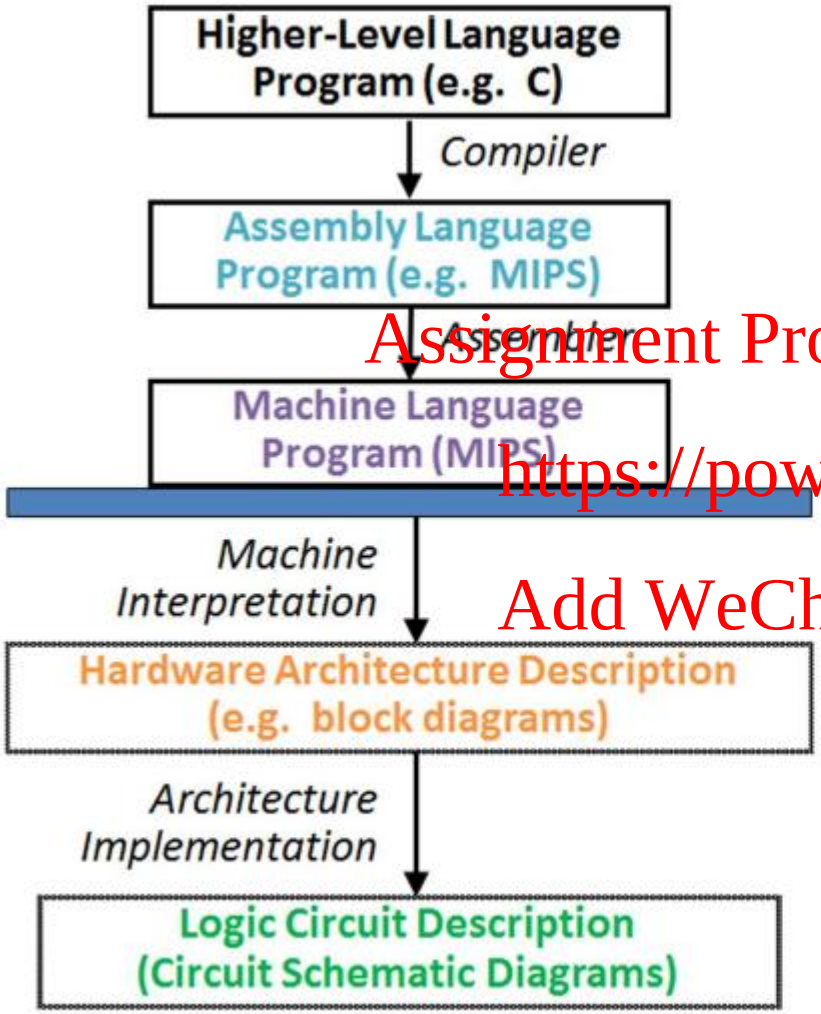
- Overview
- Assembly Language
- Machine Language
- Programming
- Addressing Modes
- The Memory Map

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Architecture: programmer's view of computer.
Defined by instructions & operand locations



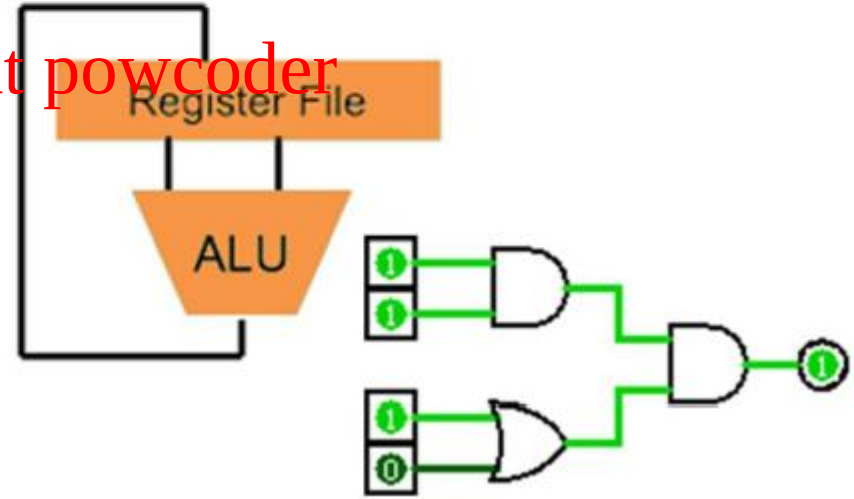
Overview




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

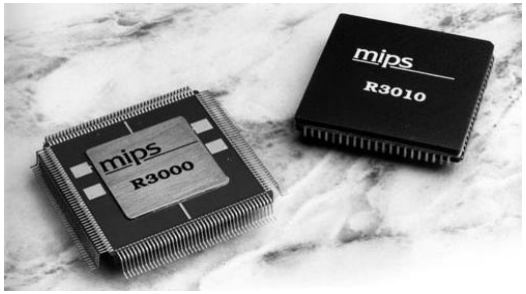
Chapter 4-1

MIPS Assembly Language

Assembly Language

- **Instructions:** commands in a computer's language
  - **Assembly language:** human-readable format of instructions
  - **Machine language:** computer-readable format (1's and 0's)
- **Instruction set:** the *vocabulary* of commands understood by a given architecture.
- **MIPS** architecture:
  - Developed by John Hennessy and his colleagues at Stanford and in the 1980's.
  - Used in many commercial systems, including Silicon Graphics, Nintendo, and Cisco
- Other instruction set architectures: Intel x86, ARM
  - Once you've learned one architecture, it's easy to learn others.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Assembly Language

- **Four design principles of computer architecture**
  - **Simplicity favors regularity**
  - **Make the common case fast**
  - **Smaller is faster**
  - **Good design demands compromise**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

MIPS Assembly Language

- Examples of core instructions sets:

Category	Instruction
Arithmetic	add (`add`), subtract (`sub`), add immediate (`addi`)
Data transfer	load word (`lw`), store word (`sw`), load byte (`lb`), store byte (`sb`), load upper immediate (`lui`)
Logical	and (`and`), or (`or`), nor (`nor`), and immediate (`andi`), or immediate (`ori`), shift left logical (`sll`), shift right logical (`srl`)
Conditional branch	branch on equal (`beq`), branch on not equal (`bne`), set on less than (`slt`)
Unconditional jump	Jump (`j`), jump register (`jr`), jump and link (`jal`)

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Addition

MIPS assembly code	C Code
`add a, b, c`	`a = b + c;`

Assignment Project Exam Help

https://powcoder.com

- add: mnemonic indicates operation to perform
- b, c: source operands (on which the operation is performed)
- a: destination operand (to which the result is written)

Add WeChat powcoder

Subtraction

- Similar to addition - only mnemonic changes

MIPS assembly code	C Code
`sub a, b, c`	`a = b - c;`

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- sub: mnemonic
- b, c: source operands
- a: destination operand

Simplicity favors regularity

- MIPS has consistent instruction format
- Same number of operands (two sources and one destination)
- Easier to encode and handle in hardware

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Multiple Instructions

- More complex code is handled by multiple MIPS instructions.

C Code	MIPS assembly code
`a = b + c - d;`	`add t, b, c  # t = b + c` `sub a, t, d  # a = t - d`

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Make the common case fast

- MIPS includes only simple, commonly used instructions
- Hardware to decode and execute instructions can be simple, small, and fast
- More complex instructions (that are less common) performed using multiple simple instructions
- MIPS is a ***reduced instruction set computer*** **(RISC)**, with a small number of simple instructions
- Other architectures, such as Intel's x86, are ***complex instruction set computers*** **(CISC)**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

CISC vs RISC

	CISC	RISC
1	Variable instruction length	Fixed instruction length
2	Large number of addressing modes	Few addressing modes.
3	Support for small number of general purpose registers.	Support for large number of general purpose registers
4	Requires less number of instructions to represent an application code when compared to RISC.	Requires more number of instructions to represent an application code when compared to CISC although this is debatable
5	Requires complex Compiler.	Requires less complex Compiler
6	Since the application code compiled for CISC instruction set results in less number of instructions we need less memory to store the application binaries in a CISC machine.	Since the application code compiled for RISC instruction set results in more number of instructions (when compared to CISC) we need more memory to store the application binaries in a RISC machine.
7	Less number of instructions need not necessarily mean that an application running on a CISC processor results in higher performance than the same running on a RISC processor.	More number of instructions need not necessarily mean that an application running on a RISC processor results in lower performance than the same running on a CISC processor
8	In addition to Load/Store there are other instructions which results in accessing memory.	Load/Store (Atomics included) are the only ones which can access memory
9	A typical CISC instruction *(Intel x 86 instruction)*.	A typical RISC instruction *(SPARC instruction)*
10	Examples of CISC processors are Intel's 486, Pentium (all flavours), AMD's Krypton, Athlon etc.	Examples of RISC processors are SUN's UltraSparc, MIPS's MIPS32, MIPS64, ARM'S ARM11, Motorola's PowerPC etc

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Operands

- Operand location: physical location in computer
  - Registers
  - Memory
  - Constants (also called *immediates*)

Assignment Project Exam Help

Location	Example	Comments
32 Registers	$t0 .. $t7 $s0 .. $s7 $sp, $ra	• Fast locations for data. • In MIPS, data must be in registers to perform arithmetic
2^{30} memory words	Mem[0] Mem[4], … Mem[4294967292]	• Accessed only by data transfer instructions (load & store). • MIPS uses byte addresses, so sequential word addresses differ by 4.

https://powcoder.com
Add WeChat powcoder

Operands: Registers

- MIPS has 32 32-bit registers
- Registers are faster than memory
- MIPS is called “32-bit architecture” because it operates on 32-bit data

Name	Register Number	Usage
`$0`	0	the constant value 0
`$at`	1	assembler temporary
`$v0-$v1`	2-3	Function return values
`$a0-$a3`	4-7	Function arguments
`$t0-$t7`	8-15	temporaries
`$s0-$s7`	16-23	saved variables
`$t8-$t9`	24-25	more temporaries
`$k0-$k1`	26-27	OS temporaries
`$gp`	28	global pointer
`$sp`	29	stack pointer
`$fp`	30	frame pointer
`$ra`	31	Function return address

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Operands: Registers

- Registers:
  - `$` before name
  - Example: `$0`, “register zero”, “dollar zero”

- Registers used for specific purposes:
  - `$0` always holds the constant value 0.
  - the *saved registers*, `$s0`–`$s7`, used to hold variables
  - the *temporary registers*, `$t0` – `$t9`, used to hold intermediate values during a larger computation
  - Discuss others later

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Smaller is Faster

- MIPS includes only a small number of registers
  - A very large number of registers may increase the clock cycle time because it takes electronic signals longer when they must travel farther.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Instructions with Registers

- Revisit add instruction

C Code

```
a = b + c
```

MIPS assembly code

```
# $s0 = a, $s1 = b, $s2 = c
add $s0, $s1, $s2
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Operands are replaced by registers.

Operands: Memory

- Too much data to fit in only 32 registers
- Store more data in memory
- Memory is large, but slow
- Commonly used variables kept in registers

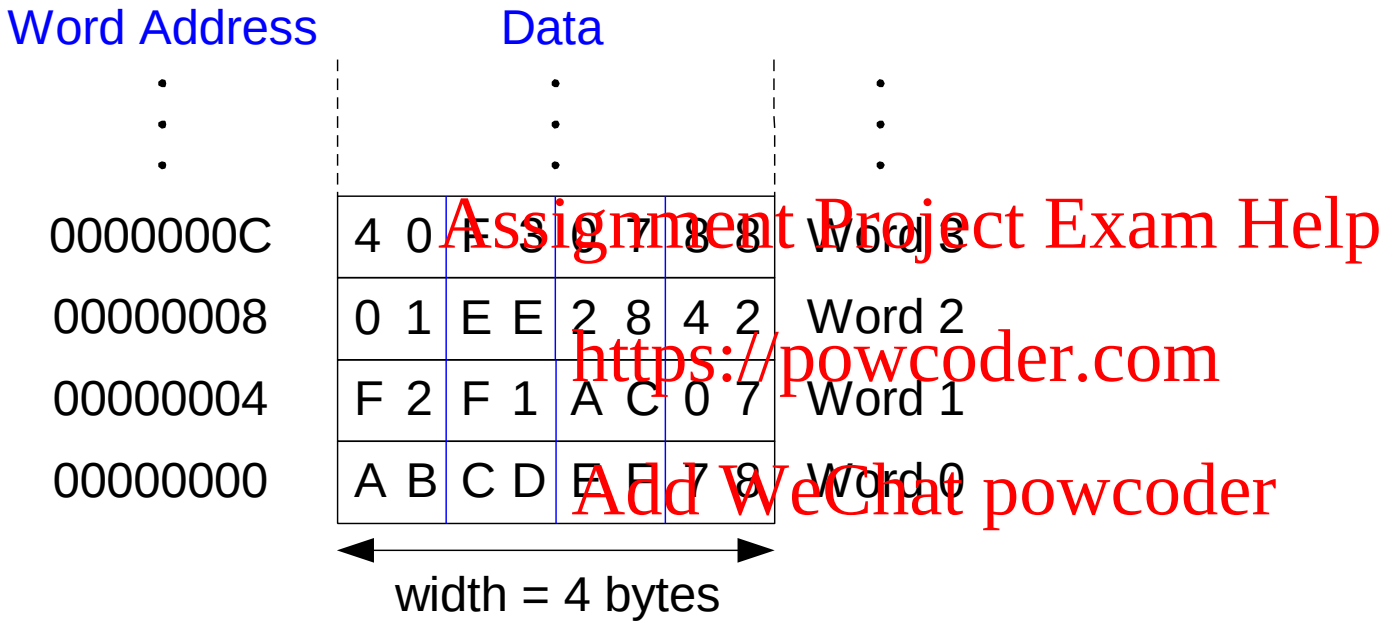
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Byte-Addressable Memory

- Each data byte has unique address
- 32-bit word = 4 bytes, so word address increments by 4



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- The address of a memory word must be multiplied by 4. For example,
  - the address of memory word 2 is 2 × 4 = 8
  - the address of memory word 10 is 10 × 4 = 40 (0x28)

Reading and Write the Memory

- Memory read is called *load*
  - Mnemonic: *load word* (`lw`)
  - Example: `lw $s0, 4($t1)`
  - Address calculation:
    - add *base address* (`$t1`) to the *offset* (4)
    - address = `($t1 + 4)`
    - Result: `$s0` holds the value at address `($t1 + 4)`
    - Any register may be used as base address
- Memory write are called *store*
  - Mnemonic: *store word* (`sw`)
  - Example: `sw $t7, 44($0)`

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Reading Byte-Addressable Memory

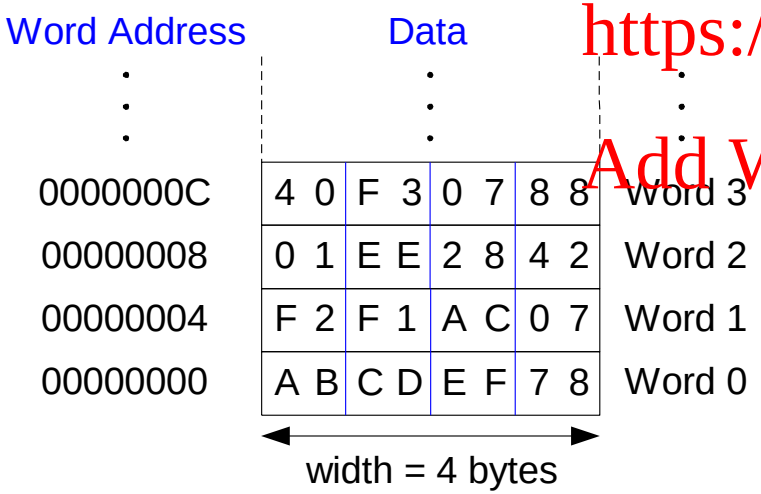
- **Example:**

  Load a word of data at memory address 4 into `$s3`.

MIPS assembly code

```
lw $s3, 4($0) # read word at address 4 into $s3
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



- `$s3` holds the value 0xF2F1 AC07 after load.

Writing Byte-Addressable Memory

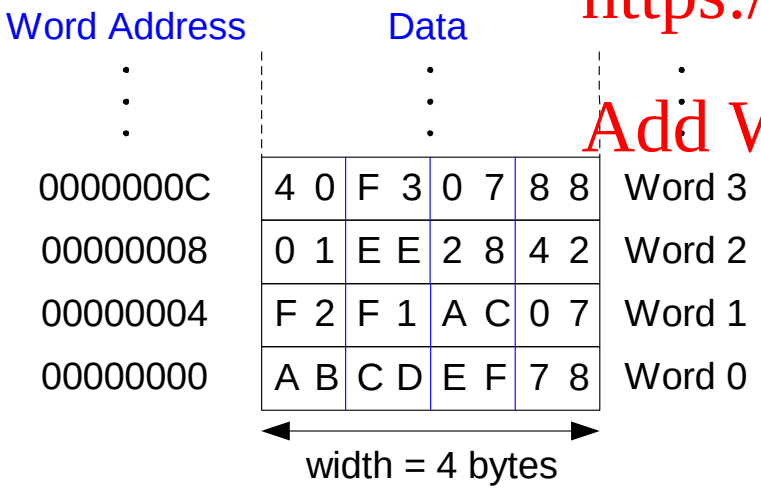
- **Example:**

  stores the value held in `$t7` into memory address 0x2C (44)

MIPS assembly code

```
sw $t7, 44($0)   # write $t7 into address 44
```

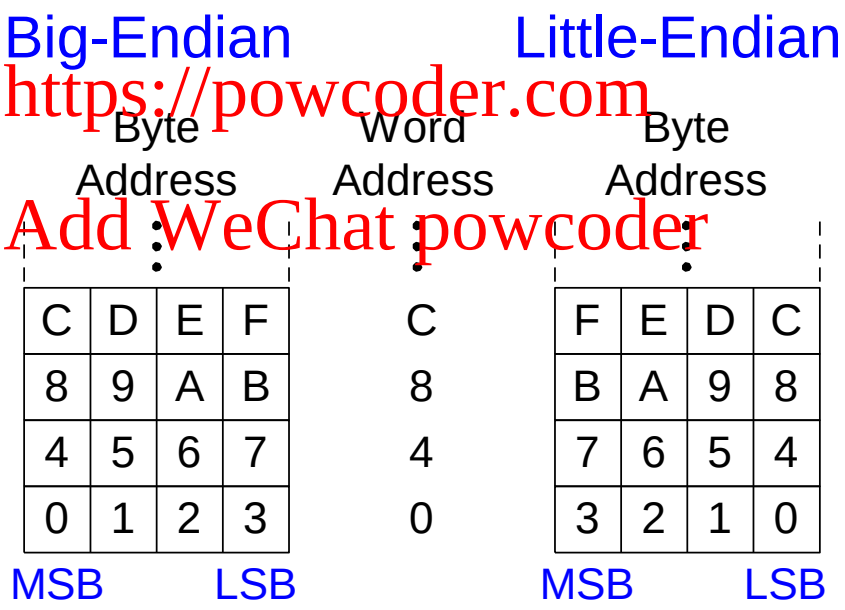
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Big-Endian & Little-Endian Memory

- How to number bytes within a word?
- Little-endian: byte numbers start at the little (least significant) end
- Big-endian: byte numbers start at the big (most significant) end
- Word address is the same for big- or little-endian

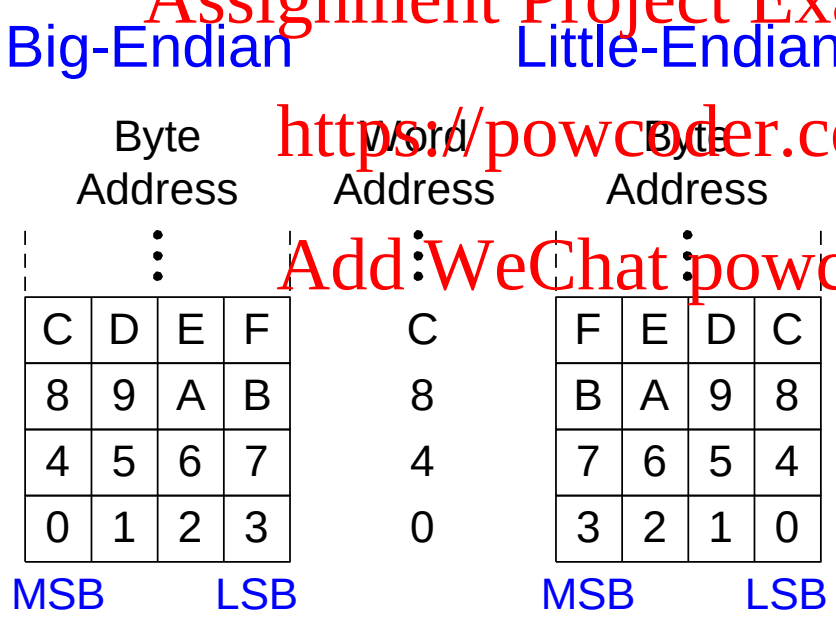
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Big-Endian & Little-Endian Memory

- Jonathan Swift's *Gulliver's Travels*: the Little-Endians broke their eggs on the little end of the egg and the Big-Endians broke their eggs on the big end
- It doesn't really matter which addressing type used – except when the two systems need to share data!

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Big-Endian & Little-Endian Example

- Suppose `$t0` initially contains `0x23456789`
- After following code runs on big-endian system, what value is `$s0`?
- In a little-endian system?

```
sw $t0, 0($0)
lb $s0, 1($0)
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Big-Endian & Little-Endian Example

- Suppose `$t0` initially contains `0x23456789`
- After following code runs on big-endian system, what value is `$s0`?
- In a little-endian system?

```
sw $t0, 0($0)
lb $s0, 1($0)
```

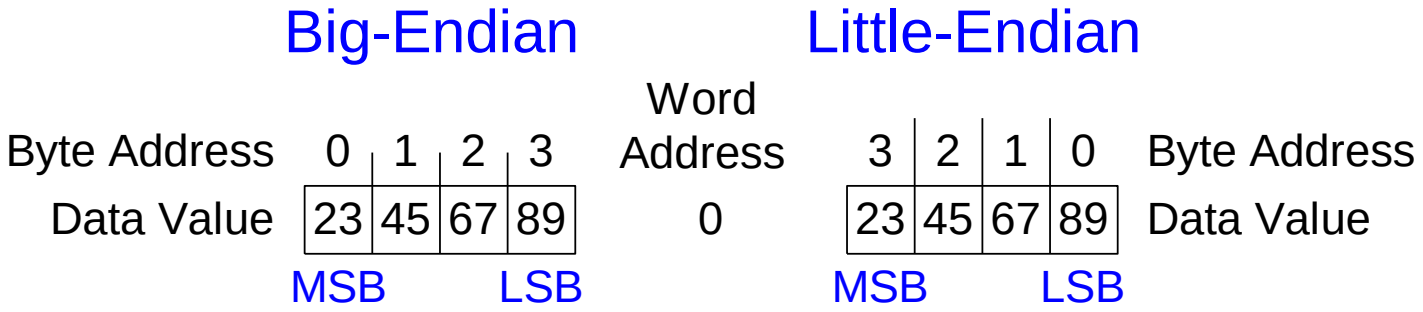
Note: MIPS is big endian

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- Answer:
  - Big-endian: 0x00000045
  - Little-endian: 0x00000067



Good design demands good compromises

- MIPS Multiple instruction formats allow flexibility
  - `add`, `sub`: use 3 register operands
  - `lw`, `sw`: use 2 register operands and a constant
- Number of instruction formats kept small
  - to adhere to design principles 1 and 3 (simplicity favors regularity and smaller is faster).

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Operands: Constants/Immediates

- `lw` and `sw` use constants or *immediates*
- *immediate*ly available from instruction
- 16-bit two's complement number
- `addi`: add immediate
- Subtract immediate (`subi`) necessary?

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

C Code	MIPS assembly code
	`# $s0 = a, $s1 = b`
`a = a + 4;`	`addi $s0, $s0, 4`
`b = a - 12;`	`addi $s1, $s0, -12`

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Chapter 4-2

Machine Language

Machine Language

- Binary representation of instructions
- Computers only understand 1's and 0's
- 32-bit instructions
  - Simplicity favors regularity: 32-bit data & instructions
- 3 instruction formats:
  - **R-Type:** register operands
  - **I-Type:** immediate operand
  - **J-Type:** for jumping (discuss later)

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Recall

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	a
11	1011	b
12	1100	c
13	1101	d
14	1110	e
15	1111	f

R-Type

- *Register-type*
- 3 register operands:
  - `rs`, `rt`: source registers
  - `rd`: destination register
- Other fields:
  - `op`: the *operation code* or `opcode` (0 for R-type instructions)
  - `funct`: the *function* with opcode, tells computer what operation to perform
  - `shamt`: the *shift amount* for shift instructions, otherwise it's 0

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

R-Type

R-Type Examples

Assembly Code

```
add $s0, $s1, $s2

sub $t0, $t3, $t5
```

Field Values

op	rs	rt	rd	shamt	funct
0	17	18	16	0	32
0	11	13	8	0	34
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Machine Code

op	rs	rt	rd	shamt	funct	
000000	10001	10010	10000	00000	100000	(0x02328020)
000000	01011	01101	01000	00000	100010	(0x016D4022)
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	

Note the order of registers in the assembly code:

```
add rd, rs, rt
```

I-Type

- *Immediate-type*
- 3 operands:
  - `rs`, `rt`: register operands
  - `imm`: 16-bit two's complement immediate
- Other fields:
  - `op`: the opcode
  - Simplicity favors regularity: all instructions have opcode
  - Operation is completely determined by opcode

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

I-Type

I-Type Examples

Assembly Code	op	rs	rt	imm
`addi $s0, $s1, 5`	8	17	16	5
`addi $t0, $s3, -12`	8	19	8	-12
`lw $t2, 32($0)`	35	0	10	32
`sw $s1, 4($t1)`	43	9	17	4
	6 bits	5 bits	5 bits	16 bits

Field Values

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Note the differing order of registers in assembly and machine codes:

```
addi rt, rs, imm
lw   rt, imm(rs)
sw   rt, imm(rs)
```

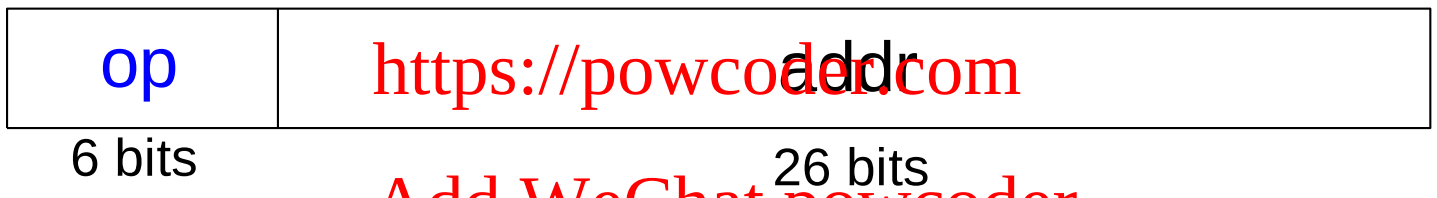
Machine Code

op	rs	rt	imm	
001000	10001	10000	0000 0000 0000 0101	(0x22300005)
001000	10011	01000	1111 1111 1111 0100	(0x2268FFF4)
100011	00000	01010	0000 0000 0010 0000	(0x8C0A0020)
101011	01001	10001	0000 0000 0000 0100	(0xAD310004)
6 bits	5 bits	5 bits	16 bits	

Machine Language: J-Type

- *Jump-type*
- 26-bit address operand (**addr**)
- Used for jump instructions (`j`)

Assignment Project Exam Help



https://powcoder.com
Add WeChat powcoder

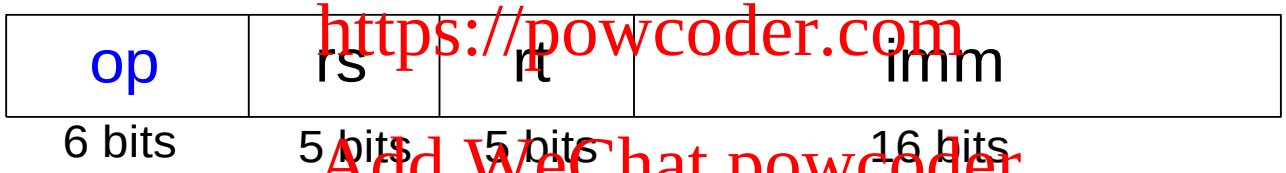
Review: Instruction Formats

R-Type

Assignment Project Exam Help

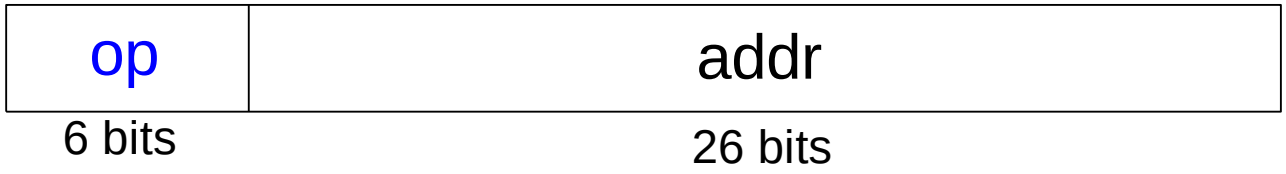
I-Type

https://powcoder.com



Add WeChat powcoder

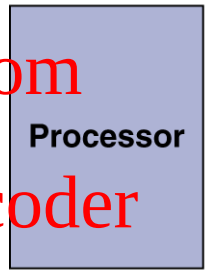
J-Type

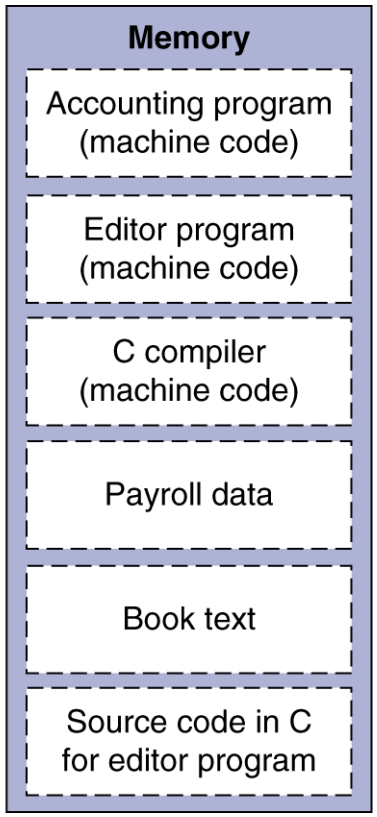


Power of the Stored Program

- Instructions represented in binary, just like data
- Instructions and data stored in memory
- Programs can operate on programs
  - e.g., compilers, linkers, …
- Binary compatibility allows compiled programs to work on different computers
  - Standardized ISAs

The BIG Picture





Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Power of the Stored Program

- 32-bit instructions & data stored in memory
- Sequence of instructions: only difference between two applications
- To run a new program:
  - No rewiring required
  - Simply store new program in memory
- Program Execution:
  - Processor *fetches* (reads) instructions from memory in sequence
  - Processor performs the specified operation

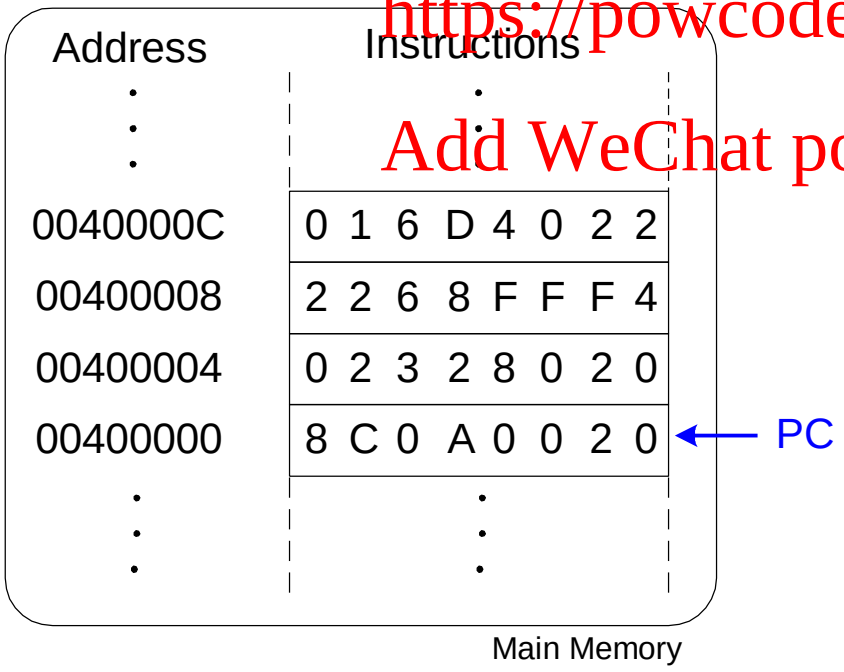
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

The Stored Program

Assembly Code	Machine Code
`lw $t2, 32($0)`	`0x8C0A0020`
`add $s0, $s1, $s2`	`0x02328020`
`addi $t0, $s3, -12`	`0x2268FFF4`
`sub $t0, $t3, $t5`	`0x016D4022`

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Stored Program



Program Counter (PC): keeps track of current instruction

Interpreting Machine Code

- Start with opcode: tells how to parse rest
- If opcode all 0's
  - R-type instruction
  - Function bits tell operation
- Otherwise
  - opcode tells operation

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder