TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG & TIN HỌC

Bài giảng

KIẾN TRÚC MÁY TÍNH

Giảng viên : Phạm Huyền Linh
Bộ môn : Toán Tin

CHƯƠNG 4

KIẾN TRÚC TẬP LỆNH

(Instruction Set Architecture)

Giảng viên: Phạm Huyền Linh

Chương 4

4.1. Tập lệnh (Instruction Set)

4.2. Kiến trúc tập lệnh CISC và RISC

4.3. Kiến trúc tập lệnh MIPS (MIPS Architecture)

4.1. Tập lệnh

- Giới thiệu chung
- Các thành phần của lệnh
- Các thao tác
- Địa chỉ toán hạng
- Định dạng lệnh

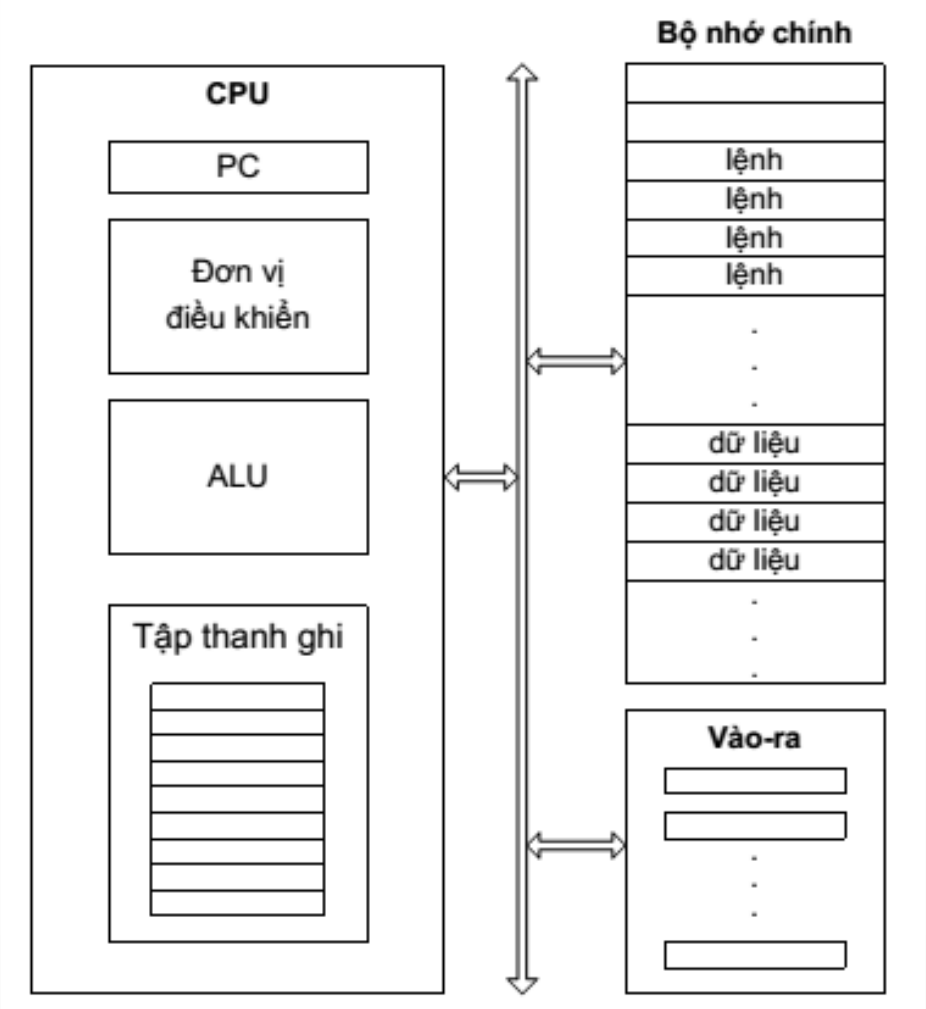
Giới thiệu chung

- Kiến trúc tập lệnh (Instruction Set Architecture)

  Cách nhìn máy tính bởi người lập trình
- Vi kiến trúc (Microarchitecture)

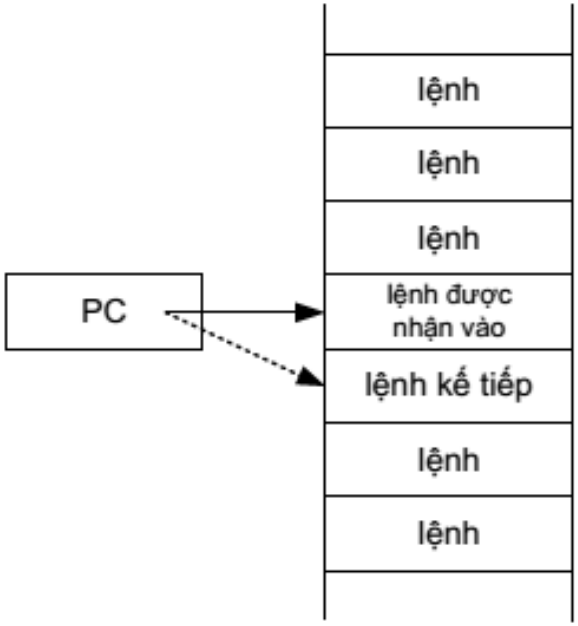
  Kiến trúc phần cứng để thực hiện lệnh
- Ngôn ngữ trong máy tính
  - Hợp ngữ(assembly language)
    - Dạng lệnh có thể đọc được bởi con người
    - Biểu diễn dạng text
  - Ngôn ngữ máy (machine language)
    - Dạng lệnh có thể đọc được bởi máy tính
    - Biểu diễn bằng các bit 0 và 1

Mô hình lập trình



Cách nhận lệnh

- PC (Program Counter)
- Lệnh được nạp vào Instruction Register -> PC tự động tăng để trỏ sang lệnh kế tiếp
- Tăng PC
  - Phụ thuộc vào chiều dài lệnh vừa nạp
  - Nếu lệnh dài 32 bit, PC tăng 4 (Địa chỉ bộ nhớ đánh theo Byte)

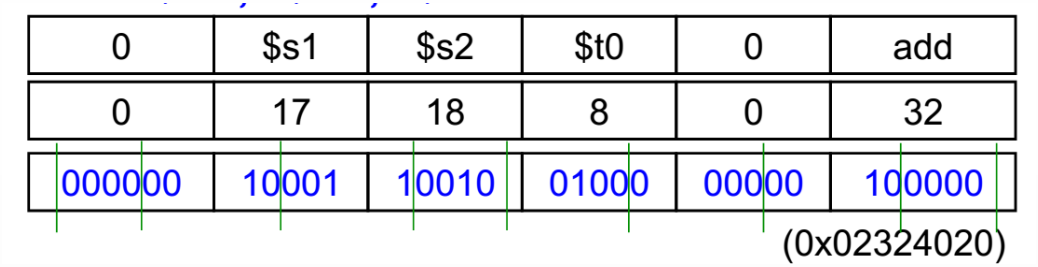


4.1. Tập lệnh

- Giới thiệu chung
- Các thành phần của lệnh
- Các thao tác
- Địa chỉ toán hạng
- Định dạng lệnh

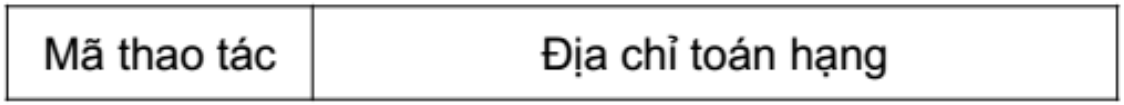
Machine Instruction

add $t0, $s1, $s2



- **Lệnh máy là các chỉ thị mà máy có thể hiểu và thực hiện được**
- **Mỗi lệnh biểu diễn bởi một chuỗi các bit nhị phân**
- **Mỗi kiến trúc vi xử lý đều có một kiến trúc tập lệnh riêng, có hàng chục đến hàng trăm lệnh**
- **Các vi xử lý khác nhau có cùng một kiến trúc tập lệnh có thể chạy chung một chương trình**
- **Mỗi một kiến trúc tập lệnh có một hợp ngữ dành riêng cho nó. Hợp ngữ mô tả các lệnh bằng các ký hiệu gợi nhớ text**

Các thành phần của lệnh



Mã thao tác (Operation Code)

Xác định thao tác cần thực hiện

Địa chỉ toán hạng

- Chỉ ra nơi chứa các toán hạng mà câu lệnh tác động
- Bao gồm:
  - Toán hạng nguồn: Có một hoặc nhiều toán hạng nguồn, trong trường hợp không có là ngầm định
  - Toán hạng đích: là nơi lưu kết quả, trong trường hợp không có là ngầm định

4.1. Tập lệnh

- Giới thiệu chung
- Các thành phần của lệnh
- Các thao tác
- Địa chỉ toán hạng
- Định dạng lệnh

Các kiểu thao tác (Types of Operations)

- Các lệnh truyền dữ liệu
- Các lệnh số học
- Các lệnh Logic
- Các lệnh vào ra
- Các lệnh điều khiển hệ thống

Các lệnh cơ bản

Type	Operation Name	Description
Data Transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
	Pop	Transfer word from top of stack to destination
Arithmetic	Add	Compute sum of two operands
	Subtract	Compute difference of two operands
	Multiply	Compute product of two operands
	Divide	Compute quotient of two operands
	Absolute	Replace operand by its absolute value
	Negate	Change sign of operand
	Increment	Add 1 to operand
	Decrement	Subtract 1 from operand

Các lệnh cơ bản

Type	Operation Name	Description
Logical	AND	Perform logical AND
	OR	Perform logical OR
	NOT (complement)	Perform logical NOT
	Exclusive-OR	Perform logical XOR
	Test	Test specified condition; set flag(s) based on outcome
	Compare	Make logical or arithmetic comparison of two or more operands; set flag(s) based on outcome
	Set Control Variables	Class of instructions to set controls for protection purposes, interrupt handling, timer control, etc.
	Shift	Left (right) shift operand, introducing constants at end
	Rotate	Left (right) shift operand, with wraparound end

Các lệnh cơ bản

Transfer of Control	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition
	Jump to Subroutine	Place current program control information in known location; jump to specified address
	Return	Replace contents of PC and other register from known location
	Execute	Fetch operand from specified location and execute as instruction; do not modify PC
	Skip	Increment PC to skip next instruction
	Skip Conditional	Test specified condition; either skip or do nothing based on condition
	Halt	Stop program execution
	Wait (hold)	Stop program execution; test specified condition repeatedly; resume execution when condition is satisfied
	No operation	No operation is performed, but program execution is continued

Các lệnh cơ bản

Input/Output	Input (read)	Transfer data from specified I/O port or device to destination (e.g., main memory or processor register)
	Output (write)	Transfer data from specified source to I/O port or device
	Start I/O	Transfer instructions to I/O processor to initiate I/O operation
	Test I/O	Transfer status information from I/O system to specified destination
Conversion	Translate	Translate values in a section of memory based on a table of correspondences
	Convert	Convert the contents of a word from one form to another (e.g., packed decimal to binary)

VD lệnh truyền DL của IBM EAS/390

Operation Mnemonic	Name	Number of Bits Transferred	Description
L	Load	32	Transfer from memory to register
LH	Load Halfword	16	Transfer from memory to register
LR	Load	32	Transfer from register to register
LER	Load (Short)	32	Transfer from floating-point register to floating-point register
LE	Load (Short)	32	Transfer from memory to floating-point register
LDR	Load (Long)	64	Transfer from floating-point register to floating-point register
LD	Load (Long)	64	Transfer from memory to floating-point register
ST	Store	32	Transfer from register to memory
STH	Store Halfword	16	Transfer from register to memory
STC	Store Character	8	Transfer from register to memory
STE	Store (Short)	32	Transfer from floating-point register to memory
STD	Store (Long)	64	Transfer from floating-point register to memory

4.1. Tập lệnh

- Giới thiệu chung
- Các thành phần của lệnh
- Các thao tác
- Địa chỉ toán hạng
- Định dạng lệnh

Địa chỉ toán hạng (Addressing Operand)

- Số lượng địa chỉ toán hạng
- Các kiểu địa chỉ toán hạng

Số lượng địa chỉ toán hạng

- 3 địa chỉ

  add r1, r2, r3 # r1 = r2 + r3

  Sử dụng phổ biến trên các kiến trúc hiện nay

- 2 địa chỉ

  add r1, r2 # r1 = r1 + r2

  Sử dụng trên Intel x86, Motorola 680x0

- 1 địa chỉ

  add r1 # Acc = Acc + r1

  Sử dụng trên kiến trúc thế hệ trước đây

- 0 địa chỉ

  add

  Cộng 2 toán hạng ở đỉnh ngăn xếp lưu vào Acc

Số lượng địa chỉ toán hạng (VD)

Instruction		Comment
SUB	Y, A, B	$Y \leftarrow A - B$
MPY	T, D, E	$T \leftarrow D \times E$
ADD	T, T, C	$T \leftarrow T + C$
DIV	Y, Y, T	$Y \leftarrow Y \div T$

(a) Three-address instructions

Instruction		Comment
MOVE	Y, A	$Y \leftarrow A$
SUB	Y, B	$Y \leftarrow Y - B$
MOVE	T, D	$T \leftarrow D$
MPY	T, E	$T \leftarrow T \times E$
ADD	T, C	$T \leftarrow T + C$
DIV	Y, T	$Y \leftarrow Y \div T$

(b) Two-address instructions

Instruction		Comment
LOAD	D	$AC \leftarrow D$
MPY	E	$AC \leftarrow AC \times E$
ADD	C	$AC \leftarrow AC + C$
STOR	Y	$Y \leftarrow AC$
LOAD	A	$AC \leftarrow A$
SUB	B	$AC \leftarrow AC - B$
DIV	Y	$AC \leftarrow AC \div Y$
STOR	Y	$Y \leftarrow AC$

(c) One-address instructions

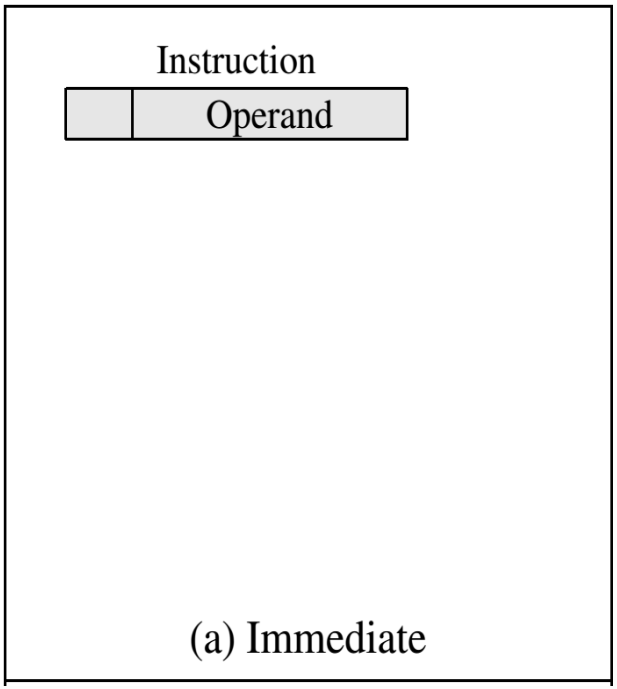
$$\text{Programs to Execute Y} = \frac{A - B}{C + (D \times E)}$$

Các kiểu địa chỉ toán hạng

- Hằng số (Immediate)
- Trực tiếp (Direct)
- Gián tiếp (Indirect)
- Thanh ghi (Register)
- Gián tiếp thanh ghi (Register Indirect)
- Displacement
- Ngăn xếp (Stack)

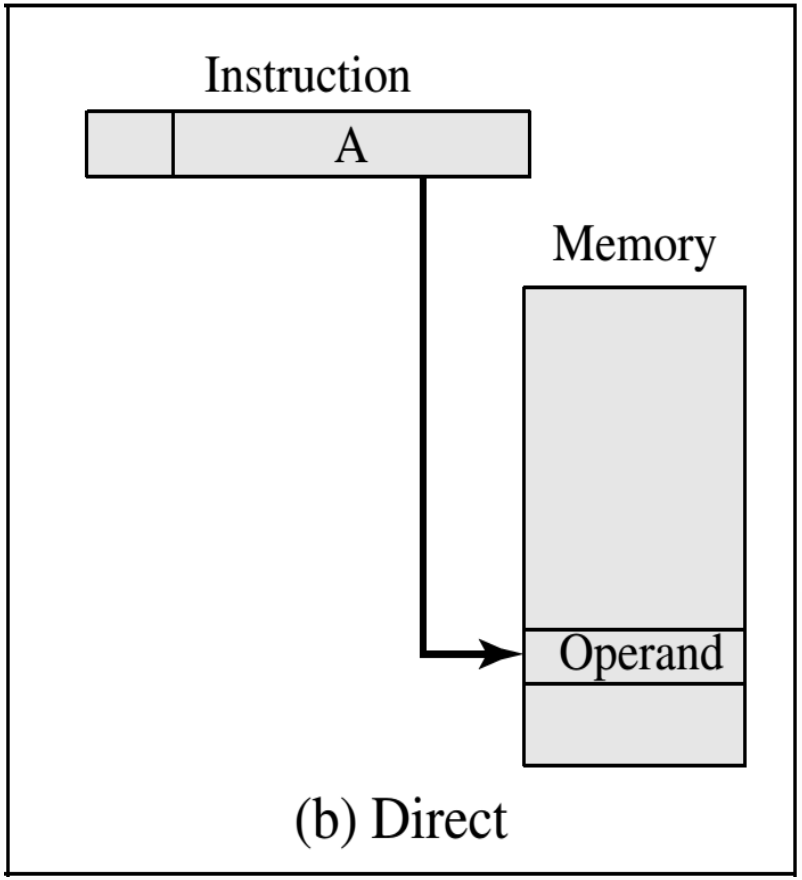
Các kiểu địa chỉ toán hạng

- **Hằng số (Immediate)**
  - **Giá trị của toán hạng nằm ngay trong câu lệnh**
  - **Có thể là:**
    - **Số có dấu: Bit đầu là bit dấu, khi nạp vào thanh ghi nó được mở rộng**
    - **Số không dấu**
  - **Tiết kiệm ô nhớ, nhanh. Tuy nhiên độ lớn của toán hạng bị hạn chế**
  - **Thường dùng nạp giá trị khởi đầu cho biến**



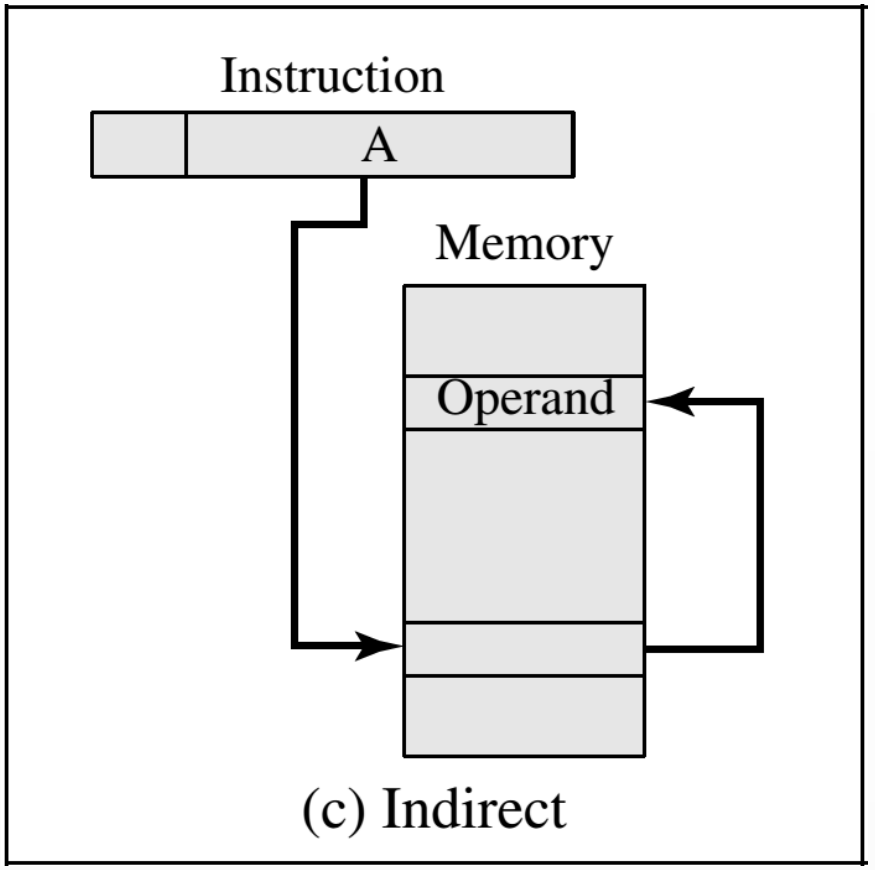
Các kiểu địa chỉ toán hạng

- **Trực tiếp (Direct)**
  - **Ít thông dụng**
  - **Bị hạn chế về không gian địa chỉ**



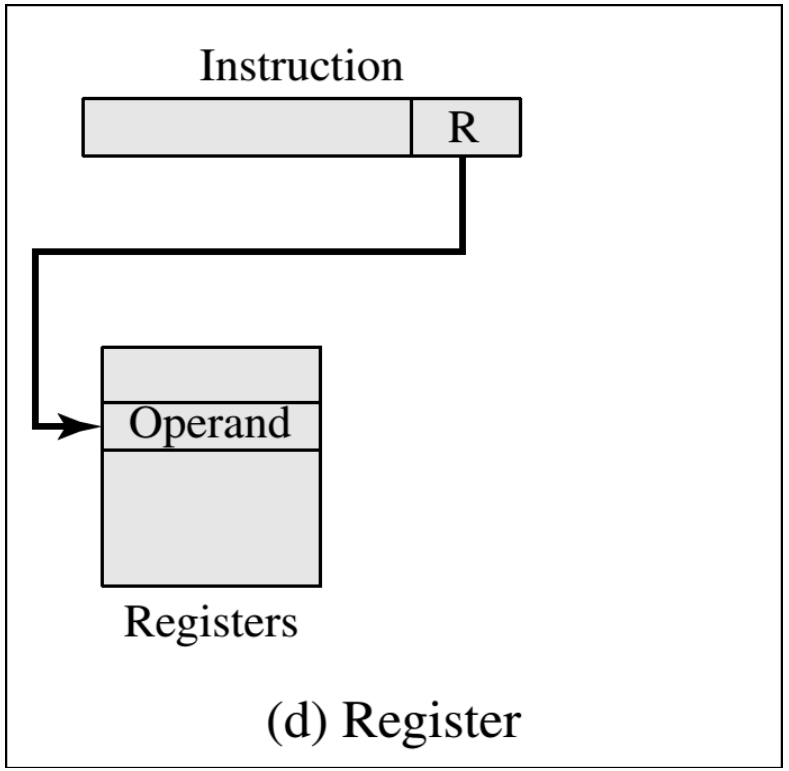
Các kiểu địa chỉ toán hạng

- **Gián tiếp (Indirect)**
  - **Không bị hạn chế không gian địa chỉ**
  - **Dùng mất 2 ô nhớ cho một toán hạng**
  - **Có thể có nhều mức gián tiếp, khi đó có một số bit dành cho việc xác định mức gián tiếp. Rất phức tạp**



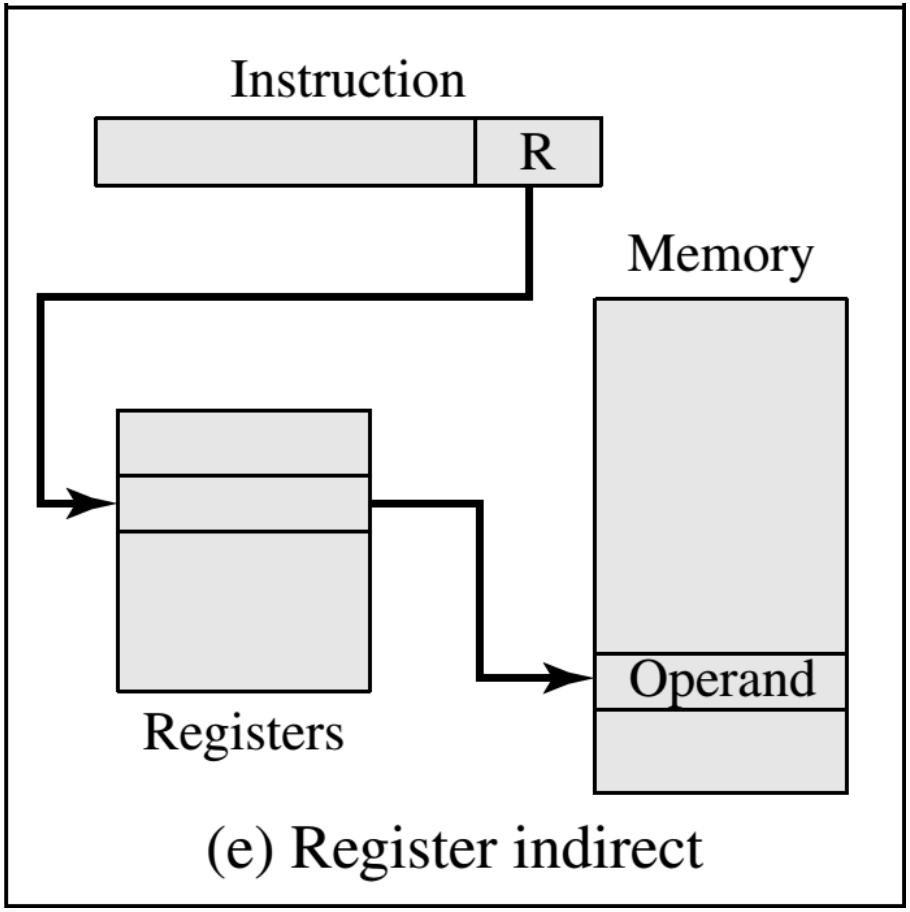
Các kiểu địa chỉ toán hạng

- **Thanh ghi (Register)**
  - Trường địa chỉ là địa chỉ của thanh ghi
  - Chiều dài trường địa chỉ thường nhỏ, 3-> 5 bit
  - Tốc độ xử lý nhanh, vì ko truy nhập bộ nhớ
  - Nhược: Số lượng thanh ghi có hạn



Các kiểu địa chỉ toán hạng

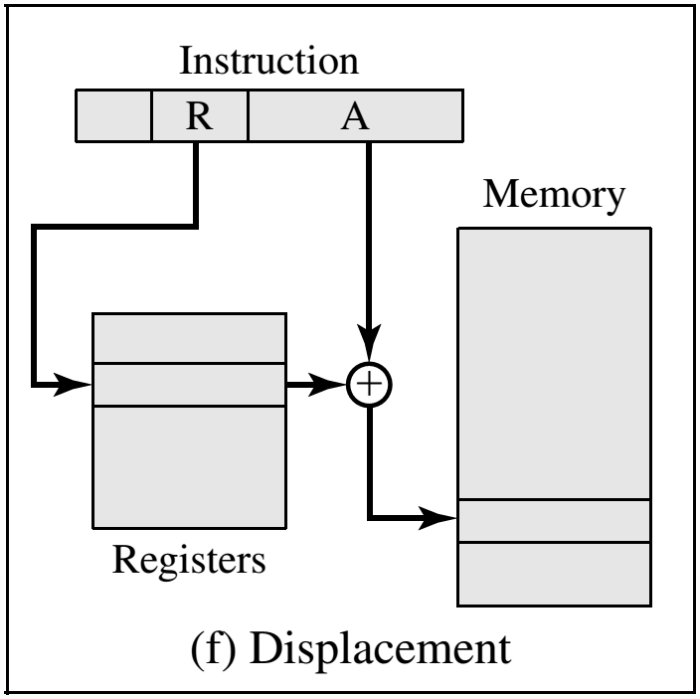
- **Gián tiếp thanh ghi (Register Indirect)**
  - **Khắc phục được hạn chế về không gian địa chỉ**
  - **Sử dụng không gian nhớ gián tiếp ít hơn kiểu địa chỉ gián tiếp**

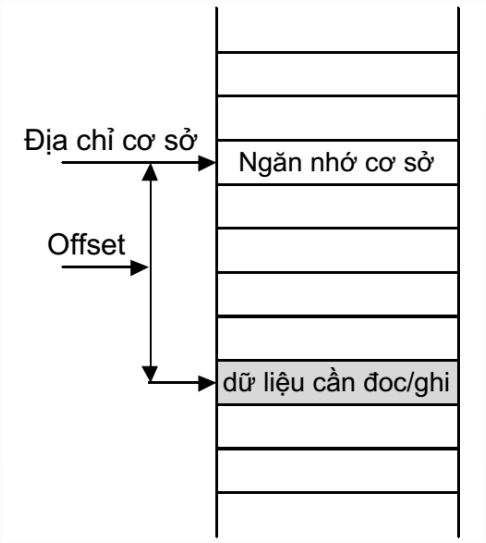



```
lw r2, 128(r3)
```

Các kiểu địa chỉ toán hạng

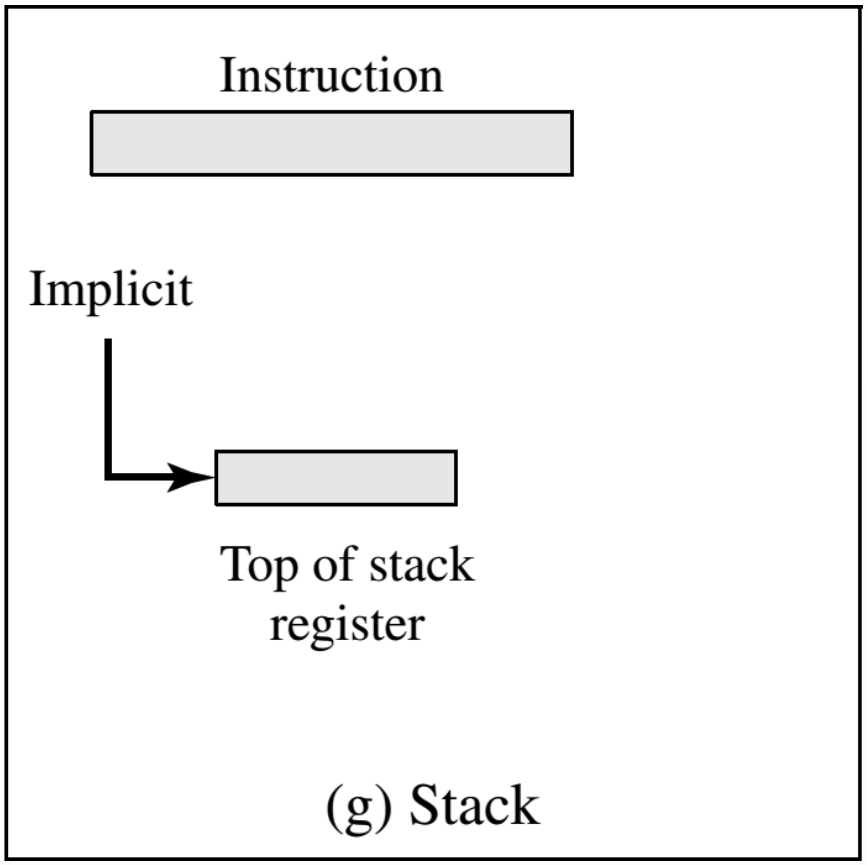
- **Displacement**
  - **Địa chỉ toán hạng= Địa chỉ cơ sở+giá trị dịch chuyển (offset)**
  - **lw r2, 128(r3)**

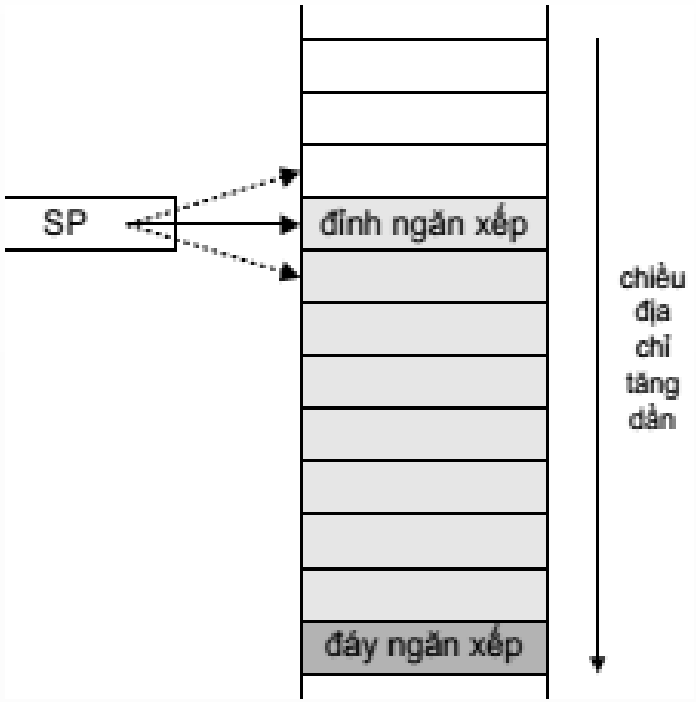




Các kiểu địa chỉ toán hạng

- **Ngăn xếp (Stack)**





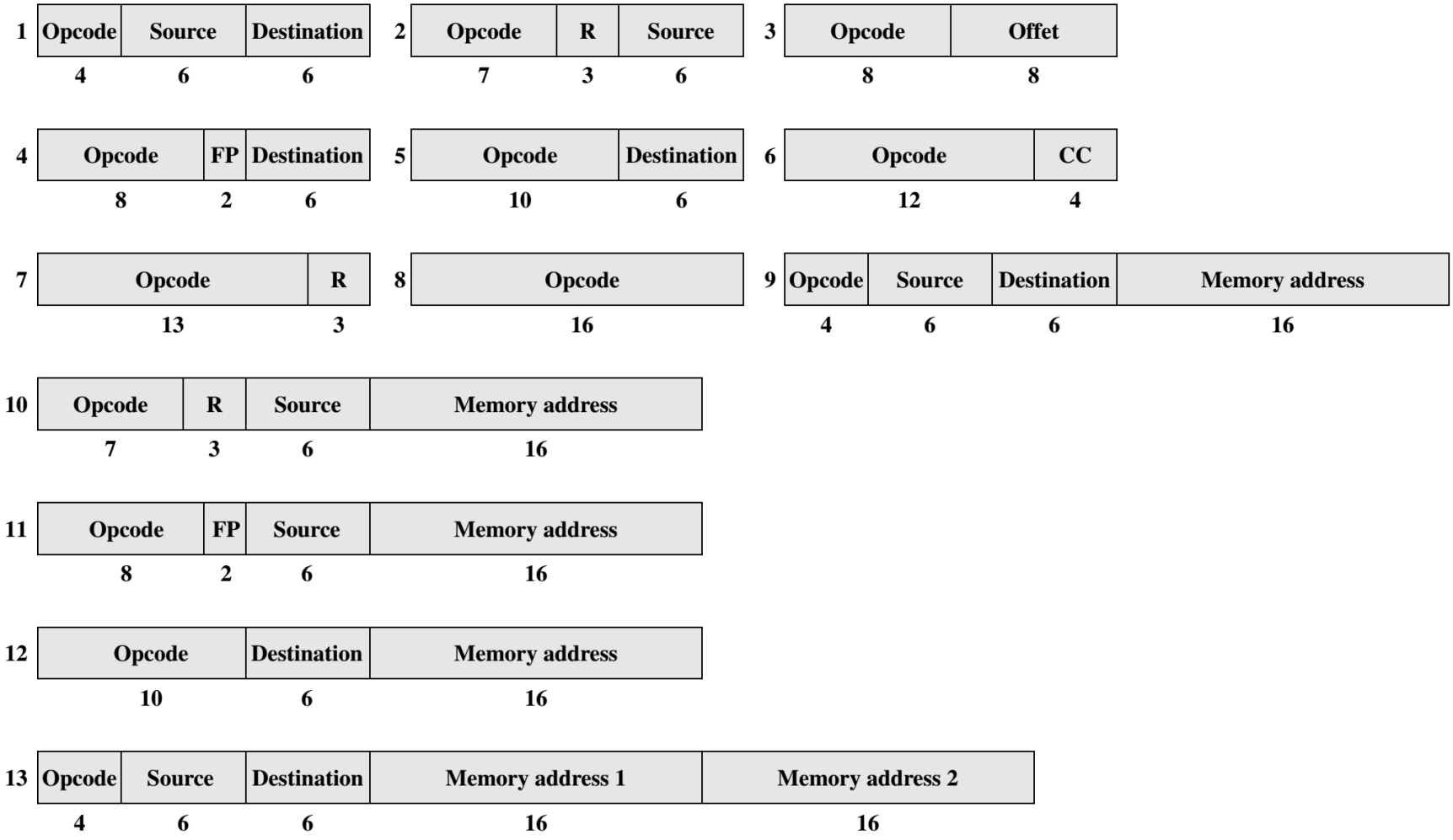
4.1. Tập lệnh

- Giới thiệu chung
- Các thành phần của lệnh
- Các thao tác
- Địa chỉ toán hạng
- Định dạng lệnh

Định dạng lệnh

- Chiều dài các lệnh
- Cách bố trí các trường trong chỉ thị lệnh
- Số bit dành cho mỗi trường
- Địa chỉ toán hạng là ẩn hay hiện
- Kiểu của địa chỉ toán hạng

Định dạng lệnh



Thiết kế tập lệnh

- Thao tác: Có bao nhiêu thao tác và độ phức tạp của các thao tác
- Kiểu DL: Kiểu DL mà các thao tác có thể thực hiện được
- Dạng của các lệnh: Chiều dài, số địa chỉ, kích thước của các trường
- Thanh ghi: Số các thanh ghi, vai trò của chúng
- Địa chỉ: Kiểu của địa chỉ

Chương 4

4.1. Tập lệnh (Instruction Set)

4.2. Kiến trúc tập lệnh CISC và RISC

4.3. Kiến trúc tập lệnh MIPS (MIPS Architecture)

Kiến trúc tập lệnh CISC và RISC

- CISC: Complex Instruction Set Computer
  - Máy tính với tập lệnh phức tạp
  - Các CPU: Intel x86, Motorola 680x0
- RISC: Reduced Instruction Set Computer
  - Máy tính với tập lệnh thu gọn
  - CPU: SunSPARC, Power PC, MIPS, ARM ...
  - RISC đối nghịch với CISC
  - Kiến trúc tập lệnh tiên tiến

Các đặc trưng của kiến trúc CISC

- Được nghĩ ra từ những năm 1960
- Lệnh có độ dài thay đổi, phức tạp gần giống với NNLT bậc cao
- Chiều dài lệnh có thể lên tới vài chăm bit
- Có nhiều chế độ địa chỉ
- Hỗ trợ các loại dữ liệu phức tạp
- Một lệnh có thể thực hiện trên nhiều chu kỳ
- Thuận tiện cho việc lập trình

Các đặc trưng của kiến trúc RISC

- Hầu hết các lệnh truy nhập toán hạng ở các thanh ghi
- Truy nhập bộ nhớ bằng các lệnh LOAD/STORE
- Thời gian thực hiện các lệnh là như nhau
- Các lệnh có độ dài cố định (thường là 32 bit)
- Số lượng dạng lệnh ít
- Có ít phương pháp định địa chỉ toán hạng
- Có nhiều thanh ghi
- Thuận tiện cho việc thiết kế song song hay tăng số lượng vi xử lý

So sánh kiến trúc CISC và RISC

- Ưu của CISC
  - Chương trình ngắn hơn
  - Thâm nhập bộ nhớ dễ dàng hơn
  - Trợ giúp mạnh hơn cho các ngôn ngữ cấp cao (Có bộ lệnh phức tạp)
  - Các tính năng có dấu phảy mạnh
- Nhược của CISC
  - Kích thước BXL lớn-> Giảm khả năng tích hợp thêm BXL
  - Tốc độ tính toán chậm
  - Thời gian xây dựng BXL lâu hơn

So sánh kiến trúc CISC và RISC

- Ưu của RISC
  - Diện tích cho BXL nhỏ ->dễ thể tích hợp thêm BXL, thanh ghi, cache
  - Tốc độ tính toán cao (Câu lệnh đơn giản, dễ giải mà, chu kỳ lệnh như nhau nên hiệu quả trong việc thực hiện kỹ thuật đường ống, không phải thâm nhập nhiều bộ nhớ)
  - Giảm chi phí thiết kế
  - Bộ điều khiển đơn giản hơn
- Nhược của RISC
  - Chương trình dài
  - Ít câu lệnh hỗ trợ ngôn ngữ bậc cao

Chương 4

4.1. Tập lệnh (Instruction Set)

4.2. Kiến trúc tập lệnh CISC và RISC

4.3. Kiến trúc tập lệnh MIPS (MIPS Architecture)

Kiến trúc tập lệnh MIPS

- MIPS viết tắt cho
  Microprocessor without Interlocked Pipeline Stages
- Được phát triển bởi John Hennessy và các đồng nghiệp ở đại học Stanford (1984)
- Được thương mại hóa bởi MIPS Technologies
- Là kiến trúc RISC điển hình, dễ học
- Được sử dụng trong nhiều sản phẩm thực tế
- Ban đầu MIPS là kiến trúc 32 bit, sau này mở rộng ra 64bit.
- MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MIPS 32 và MIPS 64. Hiện nay tồn tại MIPS 32 và MIPS 64.

Tập thanh ghi của MIPS

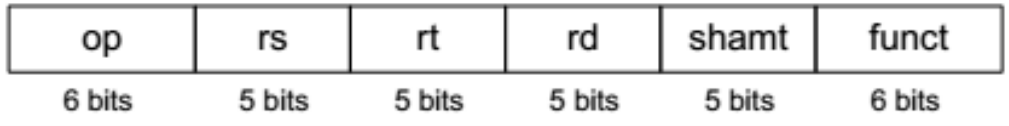
- MIPS có tập 32 thanh ghi 32-bit
- Được sửdụng thường xuyên
- Được đánh số từ 0 đến 31 (mã hóa bằng 5-bit)
- Chương trình hợp ngữ đặt tên:
- Bắt đầu bằng dấu $
- $t0, $t1, …, $t9 chứa các giá trị tạm thời
- $s0, $s1, …, $s7 cất các biến
- Qui ước gọi dữ liệu trong MIPS:
- Dữ liệu 32-bit được gọi là “word”
- Dữ liệu 16-bit được gọi là “halfword”

Tập thanh ghi của MIPS

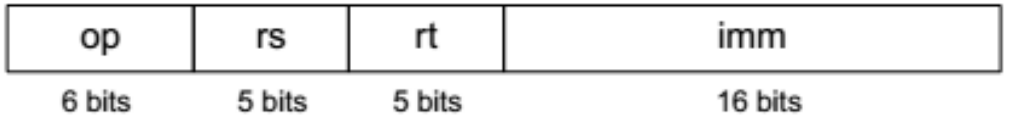
Tên thanh ghi	Số hiệu thanh ghi	Công dụng
$zero	0	the constant value 0, chứa hằng số = 0
$at	1	assembler temporary, giá trị tạm thời cho hợp ngữ
$v0-$v1	2-3	procedure return values, các giá trị trả về của thủ tục
$a0-$a3	4-7	procedure arguments, các tham số vào của thủ tục
$t0-$t7	8-15	temporaries, chứa các giá trị tạm thời
$s0-$s7	16-23	saved variables, lưu các biến
$t8-$t9	24-25	more temporarie, chứa các giá trị tạm thời
$k0-$k1	26-27	OS temporaries, các giá trị tạm thời của OS
$gp	28	global pointer, con trỏ toàn cục
$sp	29	stack pointer, con trỏ ngăn xếp
$fp	30	frame pointer, con trỏ khung
$ra	31	procedure return address, địa chỉ trở về của thủ tục

Các kiểu lệnh máy của MIPS

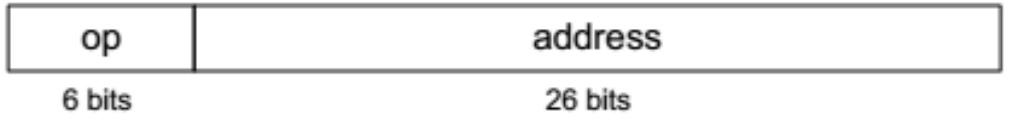
- Lệnh kiểu R



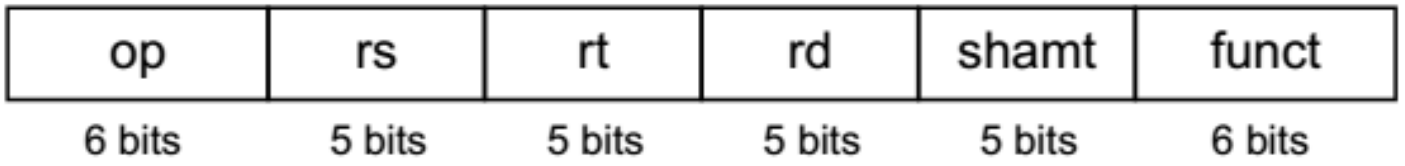
- Lệnh kiểu I



- Lệnh kiểu J



Lệnh kiểu R (Registers)



Các trường của lệnh

- **`op`** (operation code - opcode): mã thao tác
  - với các lệnh kiểu R, op = 000000
- **`rs`**: số hiệu thanh ghi nguồn thứ nhất
- **`rt`**: số hiệu thanh ghi nguồn thứ hai
- **`rd`**: số hiệu thanh ghi đích
- **`shamt`** (shift amount): số bit được dịch, chỉ dùng cho lệnh dịch bit, với các lệnh khác shamt = 00000
- **`funct`** (function code): mã hàm

Ví dụ mã máy của lệnh add, sub

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

```
add $t0, $s1, $s2
```

0	$s1	$s2	$t0	0	add
0	17	18	8	0	32
000000	10001	10010	01000	00000	100000

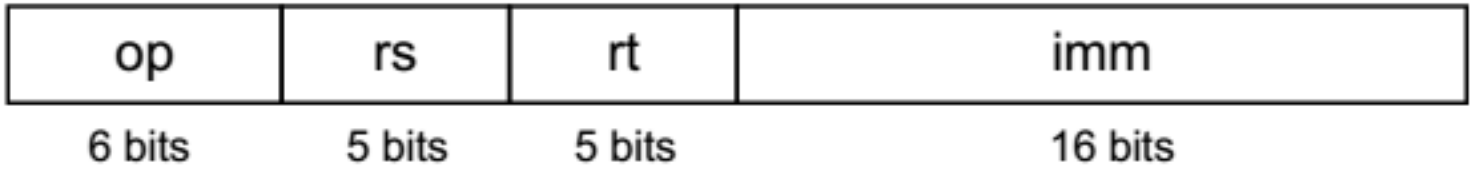
(0x02324020)

```
sub $s0, $t3, $t5
```

0	$t3	$t5	$s0	0	sub
0	11	13	16	0	34
000000	01011	01101	10000	00000	100010

(0x016D8022)

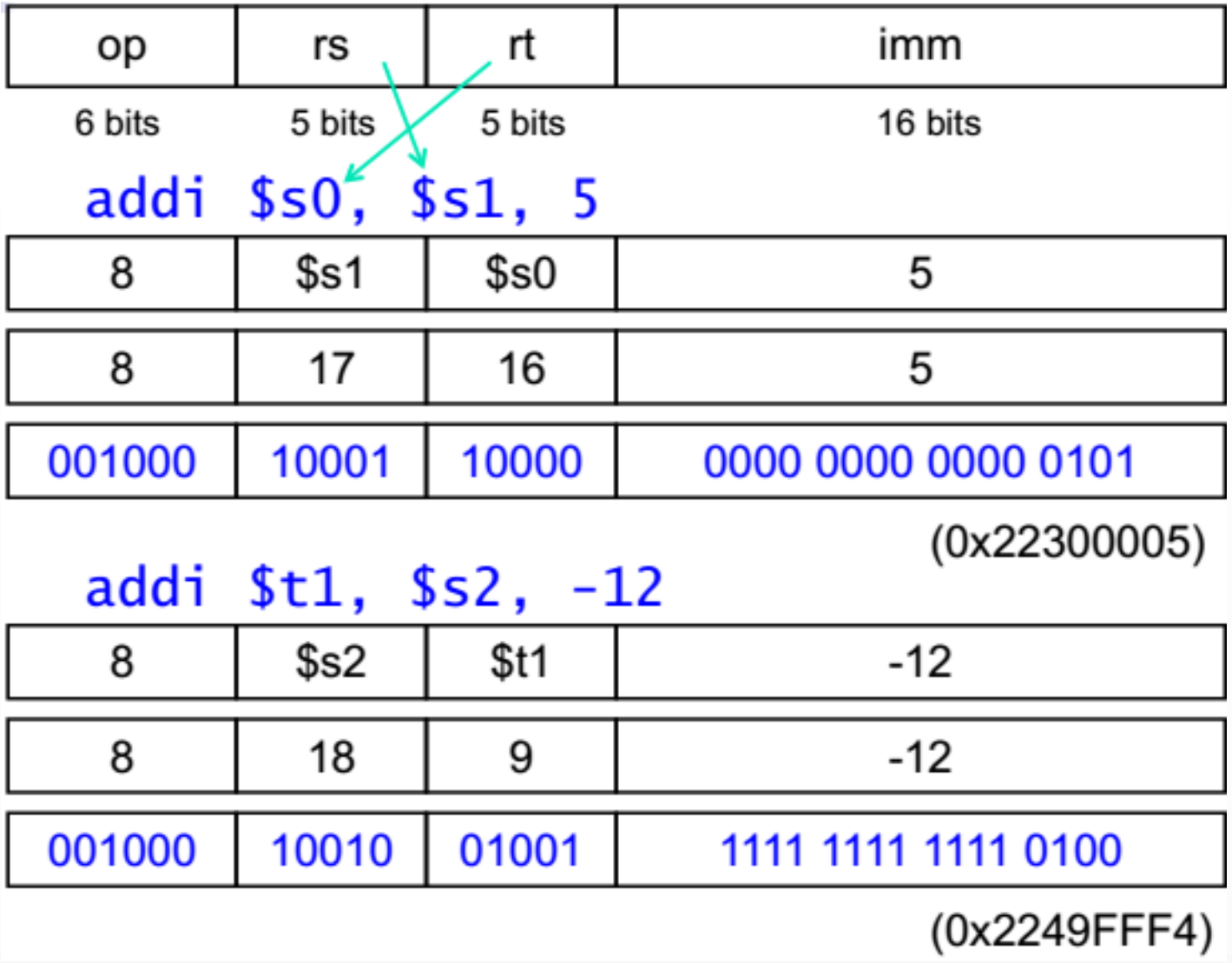
Lệnh kiểu I(Immediate)



- Dùng cho các lệnh số học/logic với toán hạng tức thì và các lệnh load/store
  - **`rs`**: số hiệu thanh ghi nguồn (addi) hoặc thanh ghi cơ sở(lw, sw)
  - **`rt`**: số hiệu thanh ghi đích (addi, lw) hoặc thanh ghi nguồn (sw)
  - `imm` (immediate): hằng số nguyên 16-bit

```
addi rt, rs, imm # (rt) = (rs)+SignExtImm
lw rt, imm(rs) # (rt) = mem[(rs)+SignExtImm]
sw rt, imm(rs) #mem[(rs)+SignExtImm] = (rt)
```

Ví dụ mã máy của lệnh addi



Ví dụ mã máy của lệnh load và store

op	rs	rt	imm
6 bits	5 bits	5 bits	16 bits

`lw $t0, 32($s3)`

35	$s3	$t0	32
35	19	8	32
100011	10011	01000	0000 0000 0010 0000

(0x8E680020)

`sw $s1, 4($t1)`

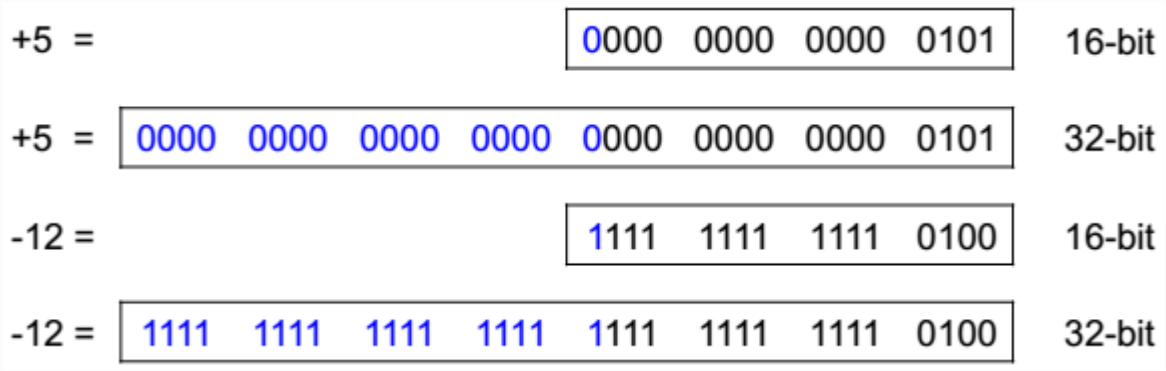
43	$t1	$s1	4
43	9	17	4
101011	01001	10001	0000 0000 0000 0100

(0xAD310004)

Mở rộng bit cho hằng số theo số có dấu

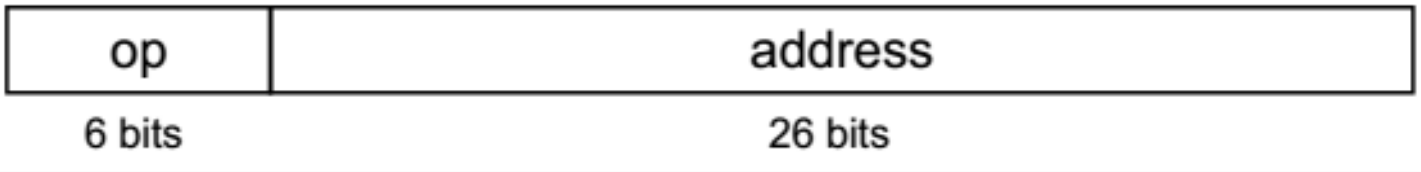
- Với các lệnh addi, lw, sw cần cộng nội dung thanh ghi với hằng số:
  - Thanh ghi có độ dài 32-bit
  - Hằng số imm 16-bit, cần mở rộng thành 32-bit theo kiểu số có dấu (Sign-extended)

- Ví dụ mở rộng số16-bit thành 32-bit theo kiểu số có dấu



Lệnh nhảy J (jump)

- Mã lệnh: op = 000010
- Toán hạng 26-bit địa chỉ
- Được sử dụng cho các lệnh nhảy
  - j (jump)
  - jal (jump and link)



VD tập lệnh MPIS-R series

ập lệnh MPIS-R series

OP	Description	OP	Description
	Load/Store Instructions	SLLV	Shift Left Logical Variable
LB	Load Byte	SRLV	Shift Right Logical Variable
LBU	Load Byte Unsigned	SRAV	Shift Right Arithmetic Variable
LH	Load Halfword		**Multiply/Divide Instructions**
LHU	Load Halfword Unsigned	MULT	Multiply
LW	Load Word	MULTU	Multiply Unsigned
LWL	Load Word Left	DIV	Divide
LWR	Load Word Right	DIVU	Divide Unsigned
SB	Store Byte	MFHI	Move From HI
SH	Store Halfword	MTHI	Move To HI
SW	Store Word	MFLO	Move From LO
SWL	Store Word Left	MTLO	Move To LO
SWR	Store Word Right		

VD tập lệnh MPIS-R series

	Arithmetic Instructions (ALU Immediate)
ADDI	Add Immediate
ADDIU	Add Immediate Unsigned
SLTI	Set on Less Than Immediate
SLTIU	Set on Less Than Immediate Unsigned
ANDI	AND Immediate
ORI	OR Immediate
XORI	Exclusive-OR Immediate
LUI	Load Upper Immediate
	Arithmetic Instructions (3-operand, R-type)
ADD	Add
ADDU	Add Unsigned
SUB	Subtract
SUBU	Subtract Unsigned
SLT	Set on Less Than
SLTU	Set on Less Than Unsigned
AND	AND
OR	OR
XOR	Exclusive-OR
NOR	NOR
	Shift Instructions
SLL	Shift Left Logical
SRL	Shift Right Logical
SRA	Shift Right Arithmetic

	Jump and Branch Instructions
J	Jump
JAL	Jump and Link
JR	Jump to Register
JALR	Jump and Link Register
BEQ	Branch on Equal
BNE	Branch on Not Equal
BLEZ	Branch on Less Than or Equal to Zero
BGTZ	Branch on Greater Than Zero
BLTZ	Branch on Less Than Zero
BGEZ	Branch on Greater Than or Equal to Zero
BLTZAL	Branch on Less Than Zero And Link
BGEZAL	Branch on Greater Than or Equal to Zero And Link
	Coprocessor Instructions
LWCz	Load Word to Coprocessor
SWCz	Store Word to Coprocessor
MTCz	Move To Coprocessor
MFCz	Move From Coprocessor
CTCz	Move Control To Coprocessor
CFCz	Move Control From Coprocessor
COPz	Coprocessor Operation
BCzT	Branch on Coprocessor z True
BCzF	Branch on Coprocessor z False
	Special Instructions
SYSCALL	System Call
BREAK	Break

HẾT CHƯƠNG 4