دانشگاه کردستان
University of Kurdistan
زانکۆی کوردستان

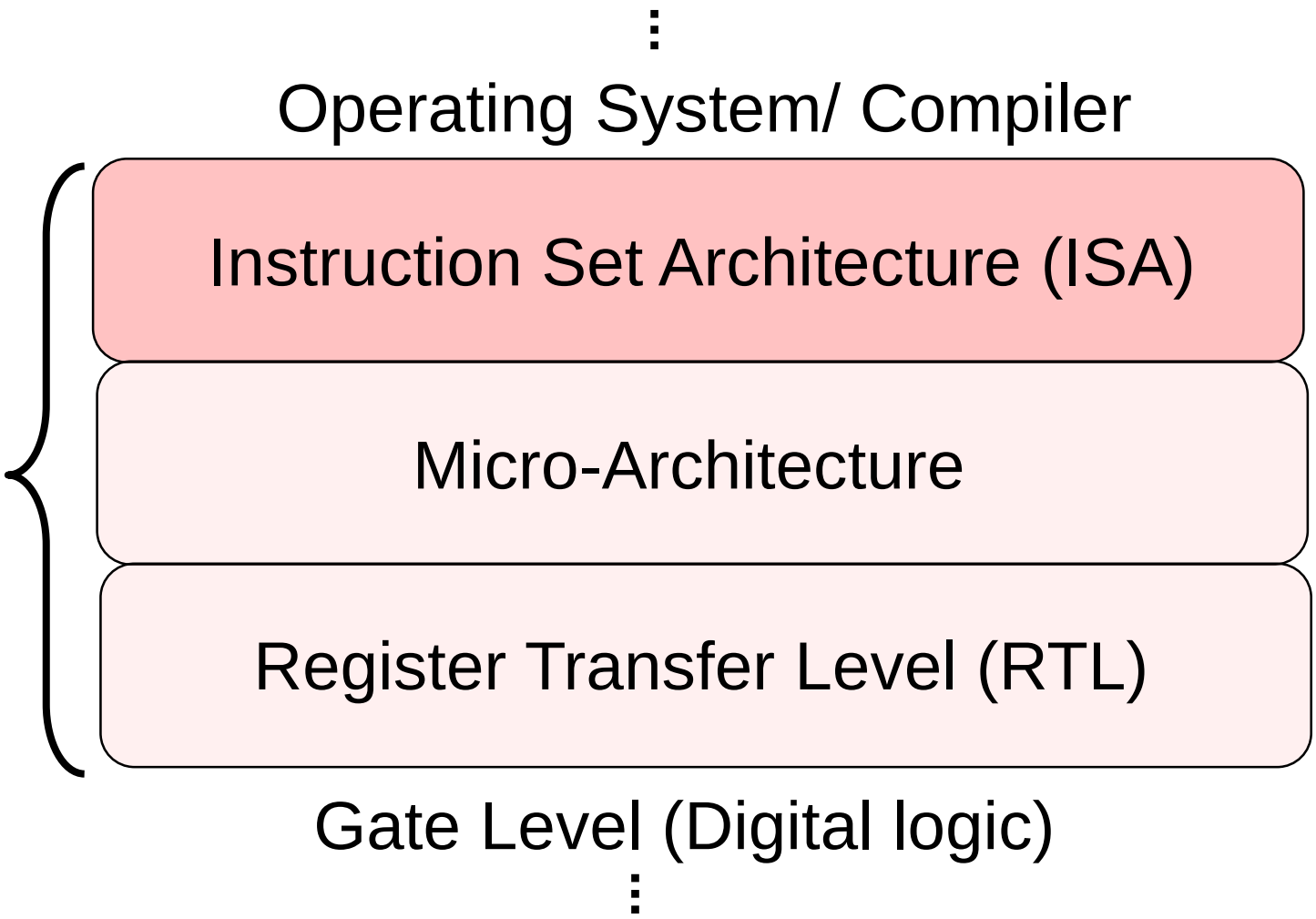
Department of Computer Engineering
University of Kurdistan

Computer Architecture

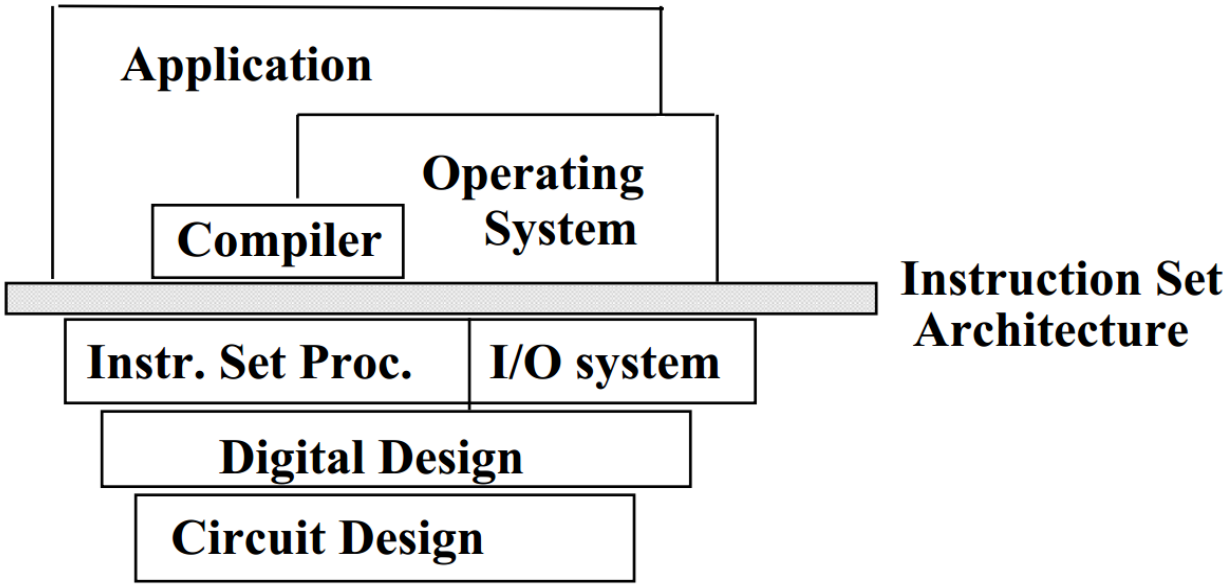
Instruction Set Architecture (ISA)

By: Dr. Alireza Abdollahpouri

لایه های معماری کامپیوتر



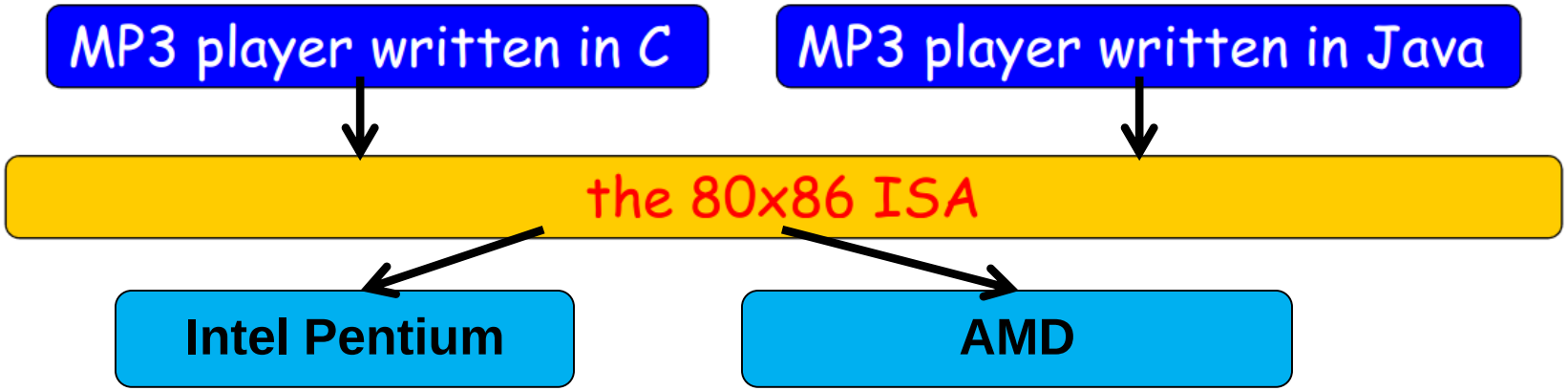
Instruction Set Architecture (ISA)



مشتمل بر تمام آن چیزی است که برنامه‌نویسان (برنامه نویسان به زبان سطح پایین و طراحان کامپایلر) برای خلق برنامه‌ای با عملکردی درست نیازمند دانستن آن‌ها هستند. همچنین مجموعه اطلاعاتی است که یک طراح پردازنده برای پیاده سازی به آن نیاز دارد. در واقع نقطه اشتراک طراحان نرم افزار و سخت افزار میباشد.

Instruction Set Architecture (ISA)

یکی از کلیدی‌ترین واسط‌های بین سخت‌افزار و نرم افزار سطح پایین، معماری مجموعه دستورالعمل (ISA) است. ISA یک سطح تجرید است که این امکان را فراهم آورده تا پیاده‌سازی‌های متعدد با قیمت و کارآیی متفاوت از یک سخت‌افزار خاص وجود داشته باشد و همه‌آن‌ها بتوانند نرم افزار واحدی را اجرا کنند.



Instruction Set Architecture (ISA)

ISA شامل موارد زیر است:

- مجموعه دستورالعملها (تعداد و نوع آنها)
- قالب دستورالعملها
- مجموعه رجیسترها
- محل ذخیره سازی عملوندها
- روشهای آدرسدهی (نحوه دسترسی به عملوندها)
- نحوه برخورد با استثناها و وقفه ها
- و تصمیمات کلی دیگر (نحوه فراخوانی زیربرنامه ها – ترتیب بایتها در حافظه و ...)

اهمیت ISA چقدر است؟

- برای استفاده کننده نهائی

  هیچ!

- برای برنامه نویس سطح بالا

  خیلی کم. تا حدی که بتواند کامپایلر مناسب را انتخاب نموده و عملکرد برنامه را بهینه کند.

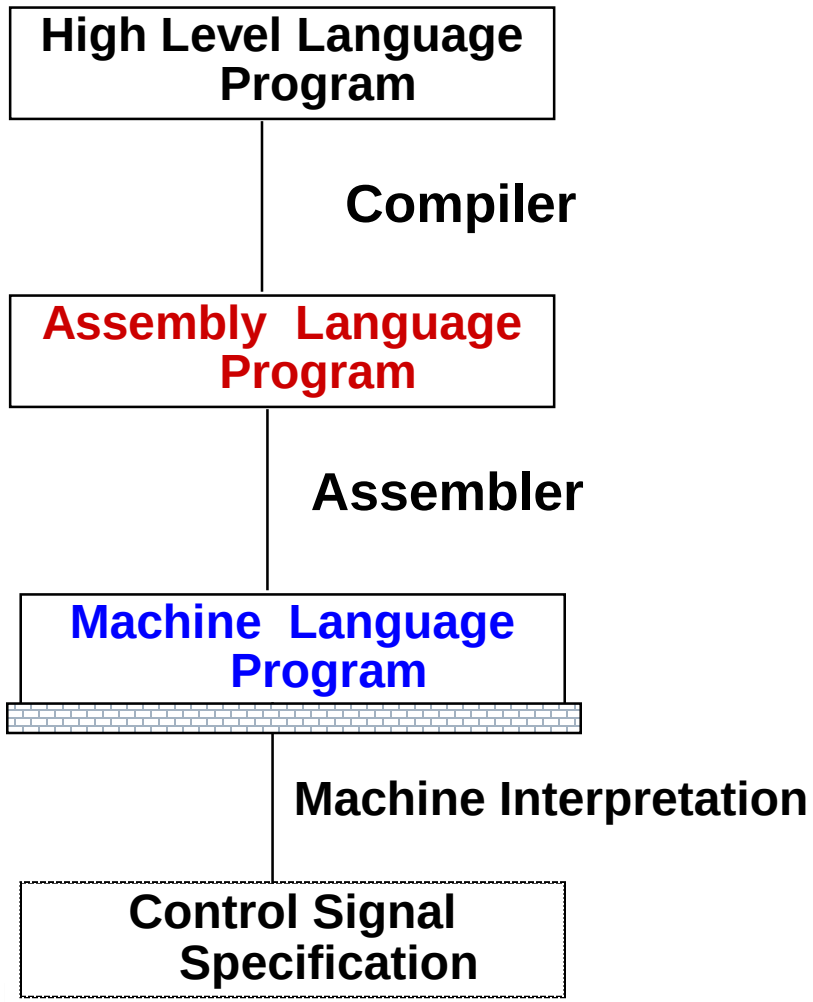
- برای برنامه نویس سطح پائین/ طراح OS

  این افراد باید اطلاعات کافی در مورد رجیسترها، ساختار حافظه، انواع داده های موجود و عملکرد دستورات داشته باشند.

- برای طراحان کامپیوتر

  این افراد نیز باید اطلاعات فوق را داشته باشند تا بتوانند اجزا سخت افزای مورد نیاز را انتخاب کنند.

How to speak to Computers?




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

زبان سطح بالا

```
lw  $15,  0($2)
lw  $16,  4($2)
sw  $16,  0($2)
sw  $15,  4($2)
```

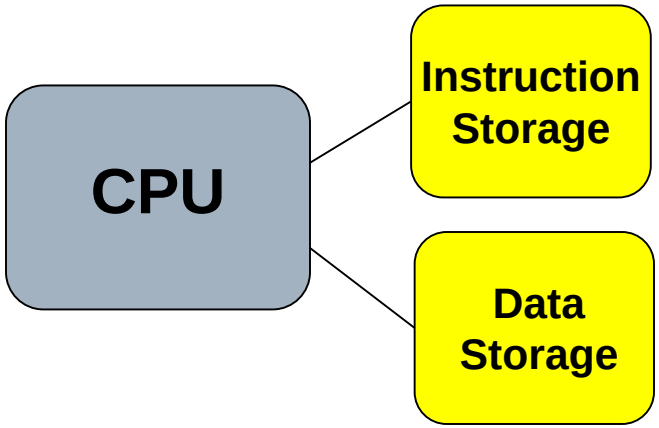
زبان اسمبلی

```
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
```

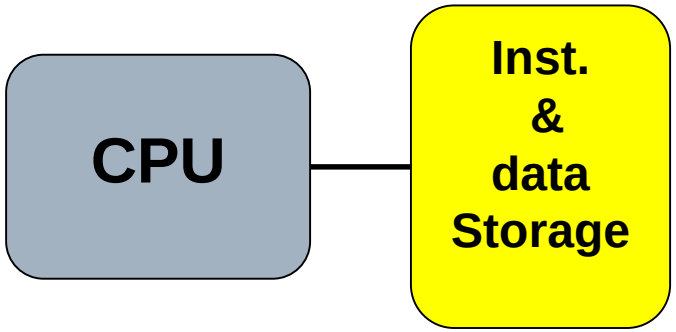
زبان ماشین

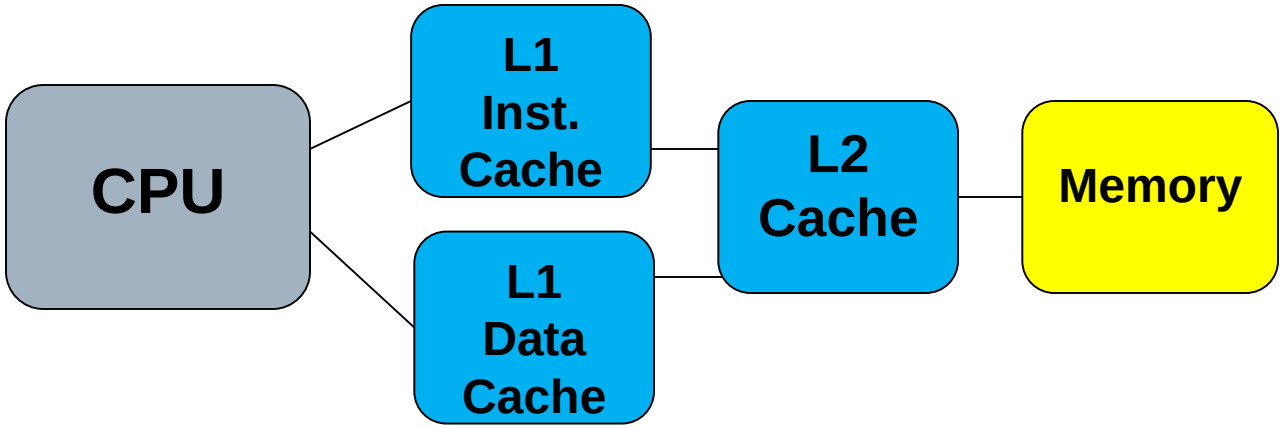
ISA- Where are the instructions?

Harvard architecture



Von-Neumann architecture

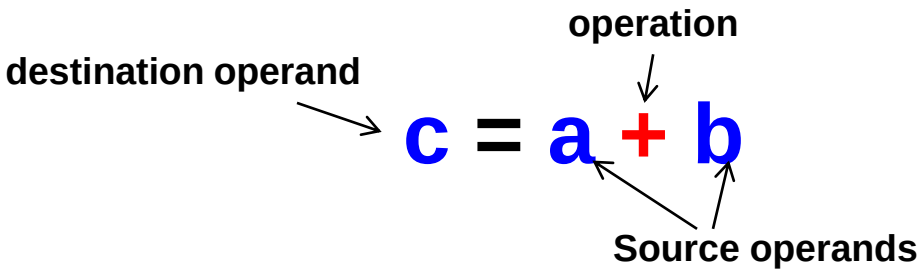




Key ISA decisions

operations

- how many?
- which ones



operands

- how many?
- location
- types
- how to specify?

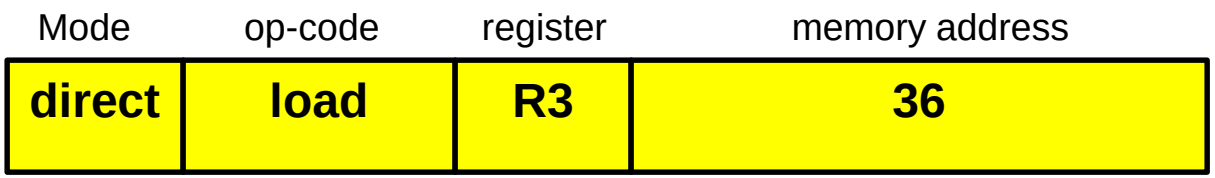
(Add R1,R2,R6)

instruction format

- size
- how many formats?

قالب دستورالعمل ها

- قالب دستورالعملها وابسته به نوع طراحی **CPU** است.
- بخش های مختلف یک دستورالعمل
  - **OP-CODE**
  - آدرس آپراندها (حافظه، ثبات ....)
  - شیوه آدرسدهی



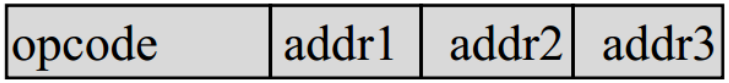
مثال:

00 0101 011 000000100100

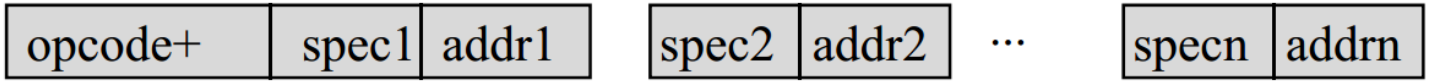
R3 ← M[36]

طول دستورالعمل

Fixed (e.g., all RISC processors -- SPARC, MIPS, Alpha) طول دستور ثابت است



Variable (VAX, ...) طول دستور متغیراست



در مورد مزایا و معایب هر کدام بحث کنید

مجموعه دستورات کامپیوتر

- دستورات مورد استفاده درکامپیوترهای مختلف از لحاظ تعداد، عملکرد، نشانه های مورد استفاده برای اسمبلی، و کد باینری بسیار متفاوت هستند با این وجود تمامی آنها دارای دستوراتی از گروههای زیر میباشند:

– دستورات انتقال داده

(Load- Store- Move- Input- Output- Push- Pop)

– دستورات محاسباتی، منطقی و جابجائی

(Add- Sub- Mul- Div- Inc – Dec- Com- And- Xor – Rotate- shift)

– دستورات کنترل برنامه

(Branch- Jump- Skip- Call- Ret)

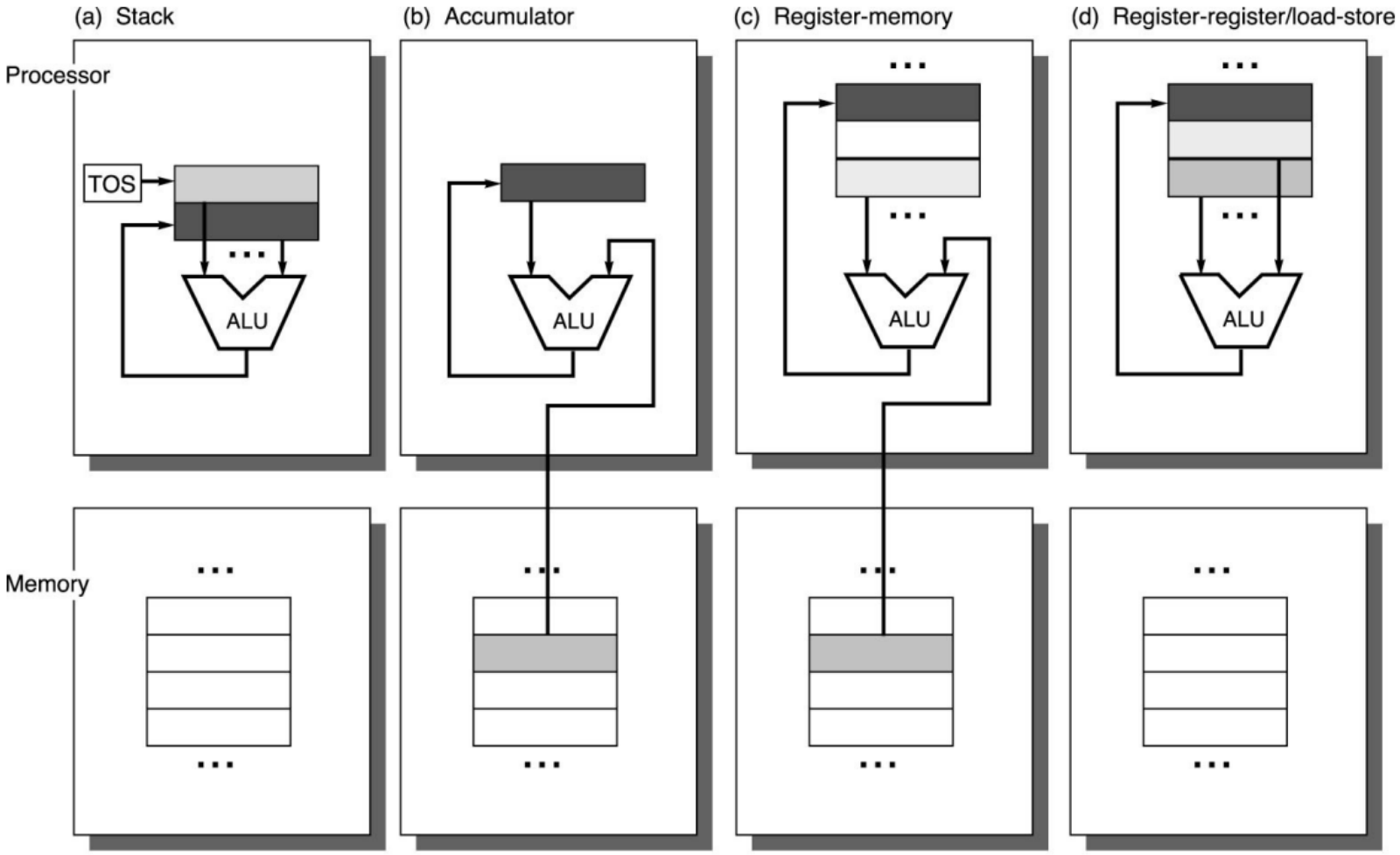
Operand Location (محل عملوندها)

- Accumulator (انباره)
- Stack (پشته)
- Registers (ثباتها)
- Memory (حافظه)

We can classify most machines into 4 types:

1- accumulator, 2- stack, 3- register-memory (most operands can be registers or memory) , 4- load-store (arithmetic operations must have register operands).

Operand Location

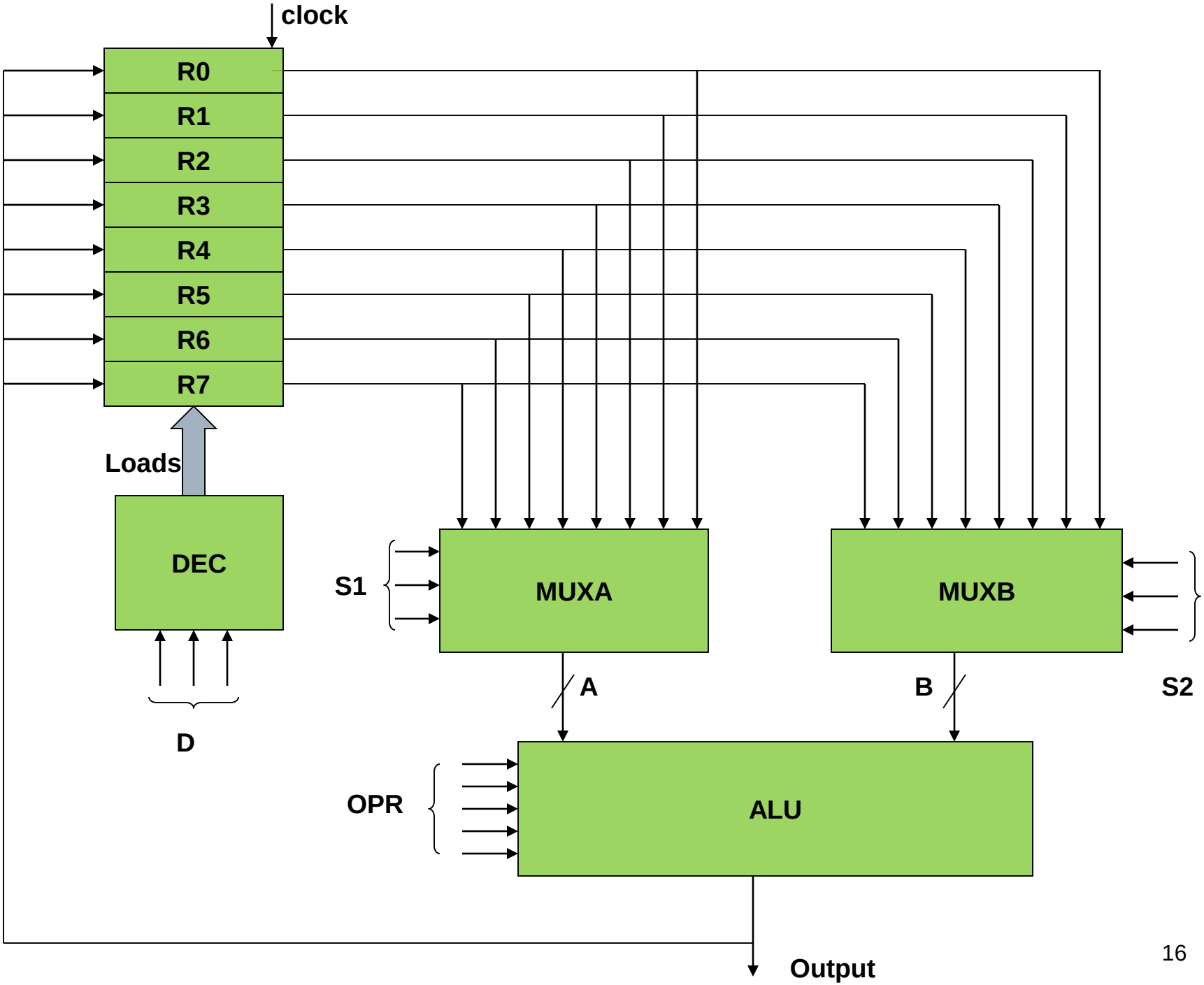


سازمان رجیستر عمومی

- General Registers

- تمامی رجیسترها می توانند به عنوان اپراند دستورالعملهای ALU بکار برده شوند.

- مثال: رجیسترهای پردازنده MIPS


clock
R0
R1
R2
R3
R4
R5
R6
R7
Loads
DEC
D
S1
MUXA
MUXB
S2
A
B
OPR
ALU
Output

مجموعه رجیسترهای عمومی و عملیات ALU

در شکل قبل:

- خروجی هر رجیستر به دو MUX متصل شده است. اینکار باعث میشود تا هر یک از آنان را بتوان آزادانه بعنوان مبدا عملیات ALU انتخاب نمود.
- برای اینکه بتوان خروجی ALU را به هر یک از رجیستر ها منتقل نمود این خروجی به ورودی تمام رجیسترها متصل شده و علاوه بر آن با استفاده از یک دیکودر مقصد عملیات را مشخص میکنیم.
- یک ALUممکن است که قادر به انجام عملیات مختلفی باشد. برای انتخاب یک عمل مورد نیاز از خطوط کنترلی OPR استفاده میشود.

مثالی از عملیات ALU

برای مثال فرض کنید که میخواهیم عملیات زیر را انجام دهیم:

$$R1 \leftarrow R2 + R3$$

برای انجام این عمل واحد کنترل باید سیگنالهای لازم را برای انتخاب ورودیهای متناسب دیکودر ,MUXA, MUXB و ALU انتخاب نماید:

۱. تعیین مقدار مناسب برای ورودی MUXA یعنی S1 طوری که محتوی رجیستر R2در روی باس A قرار گیرد.
۲. تعیین مقدار مناسب برای ورودی MUXB یعنی S2 طوری که محتوی رجیستر R3در روی باس B قرار گیرد.
۳. تعیین مقدار لازم برای ورودی OPRکه ALU را وادار به انجام عمل جمع A+B نماید.
۴. در نهایت انتخاب مقدار مناسب برای دیکور D به نحویکه خروجی ALU را به رجیستر R1 منتقل نماید.

سازمان پشته یا Stack

- ساختار LIFO
- PUSH
- POP
- دسترسی فقط به عنصر بالای پشته امکانپذیر است
- فقط یک اشاره گر نیاز داریم SP = STACK POINTER

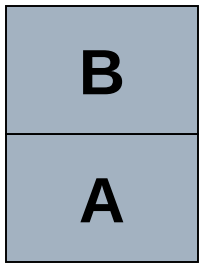
AB*CD/+

- PUSH A
- PUSH B
- MUL (POP,POP, PUSH A*B)
- PUSH C
- PUSH D
- DIV (POP,POP,PUSH C/D)
- ADD (POP,POP, PUSH RESULT)

AB*CD/+

PUSH A

PUSH B



AB*CD/+

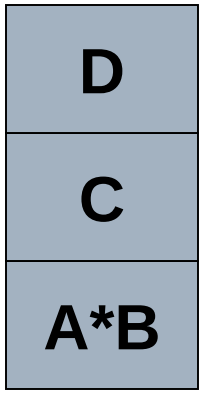
PUSH A

PUSH B

MUL

PUSH C

PUSH D



AB*CD/+

PUSH A

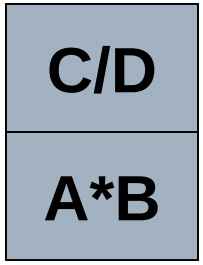
PUSH B

MUL

PUSH C

PUSH D

DIV



AB*CD/+

PUSH A

PUSH B

MUL

PUSH C

PUSH D

DIV

ADD

A*B+ C/D

تاثیر ISA بر تعداد دستورالعملها (Instruction Count)

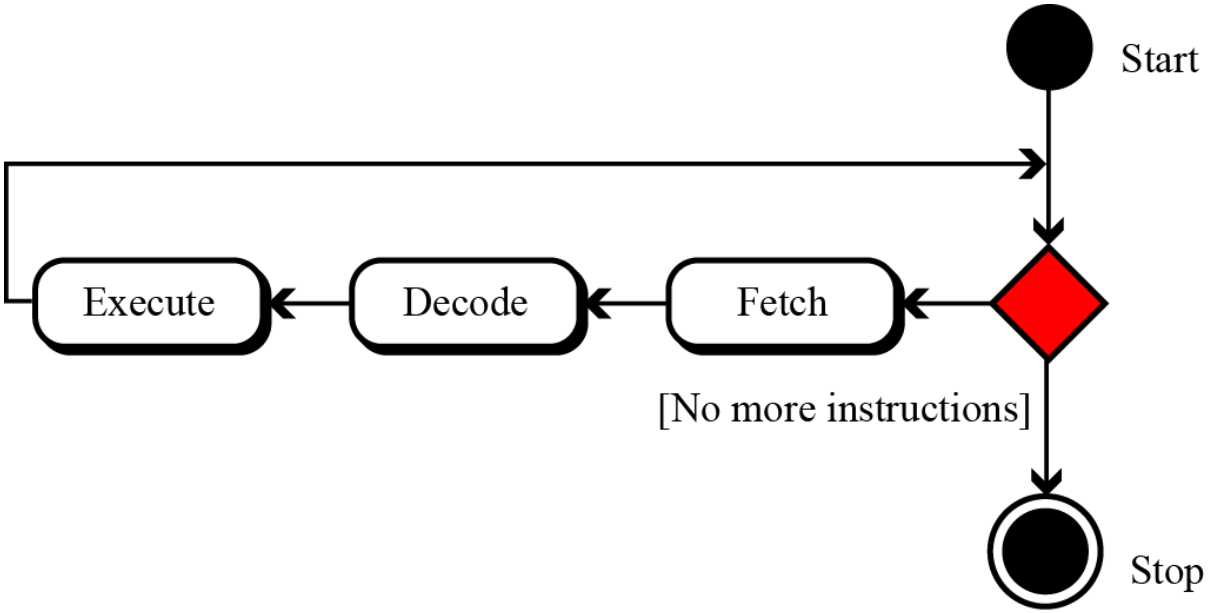
Code sequence for A = B+C for four classes of instruction sets:

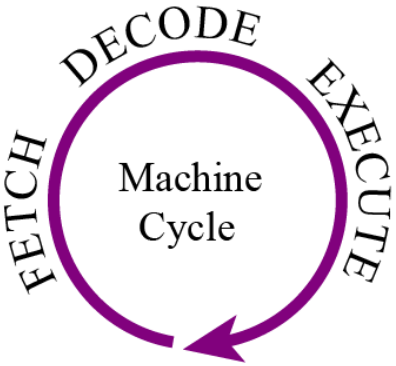
Accumulator	Stack	Register (Register-memory)	Register (Load-store)
load B	push B	load R1,B	Load R1, B
add C	push C	add R1,C	Load R2,C
store A	add	store A,R1	Add R1,R1,R2
	pop A		Store A,R1

پردازنده MIPS از این نوع است

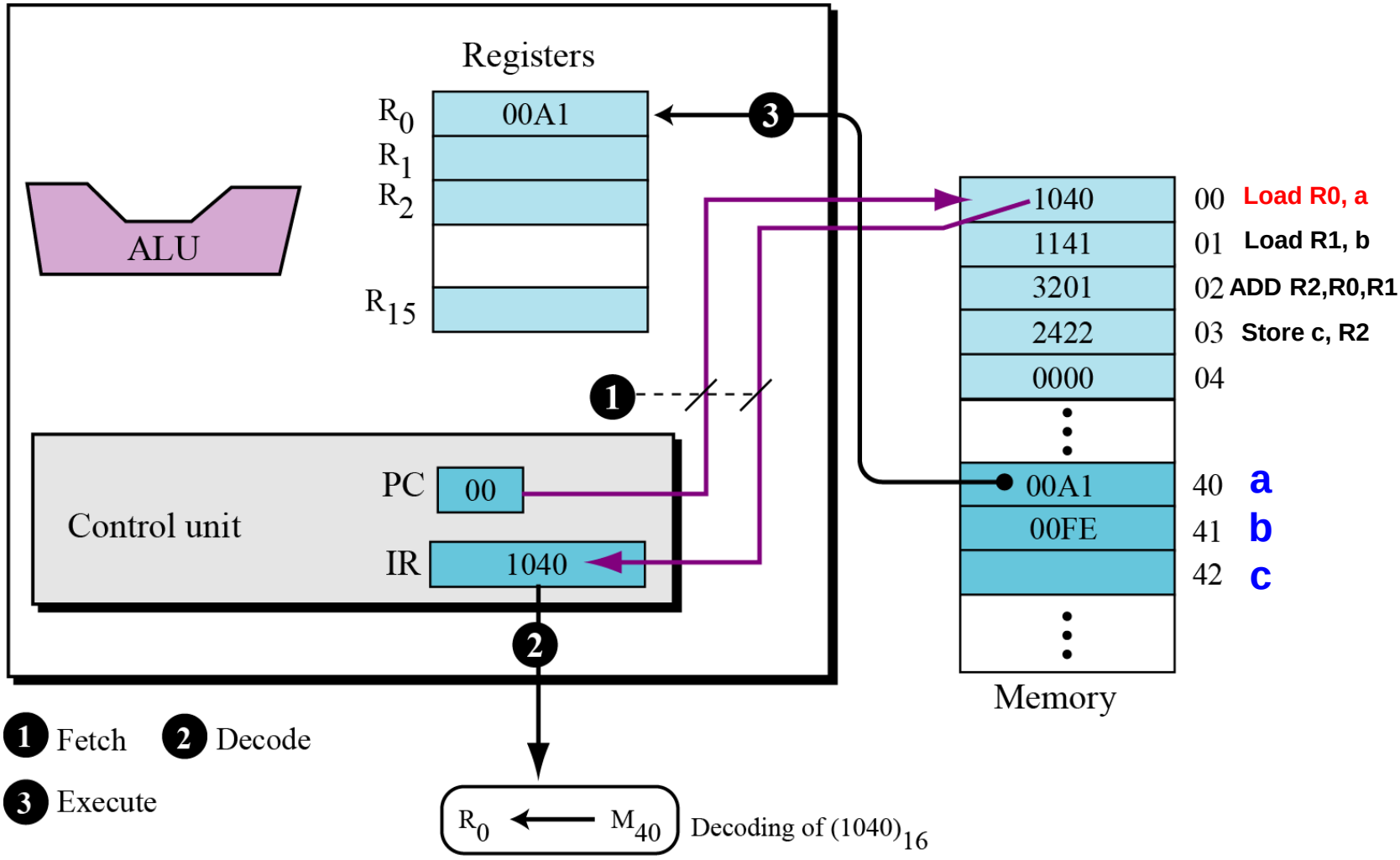
Instruction cycle

The CPU uses repeating machine cycles to execute instructions in the program, one by one, from beginning to end. A simplified cycle can consist of three phases: fetch, decode and execute

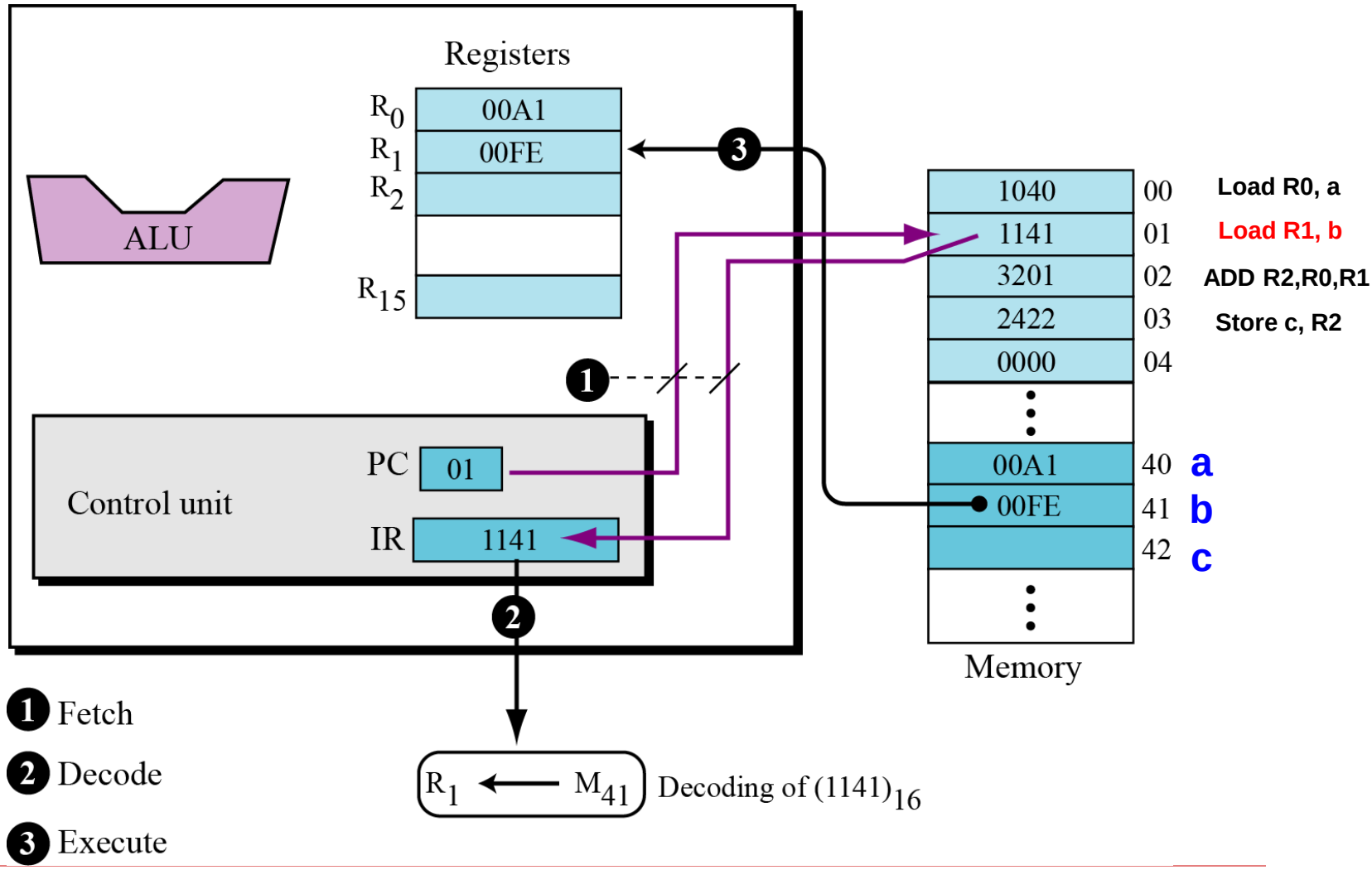




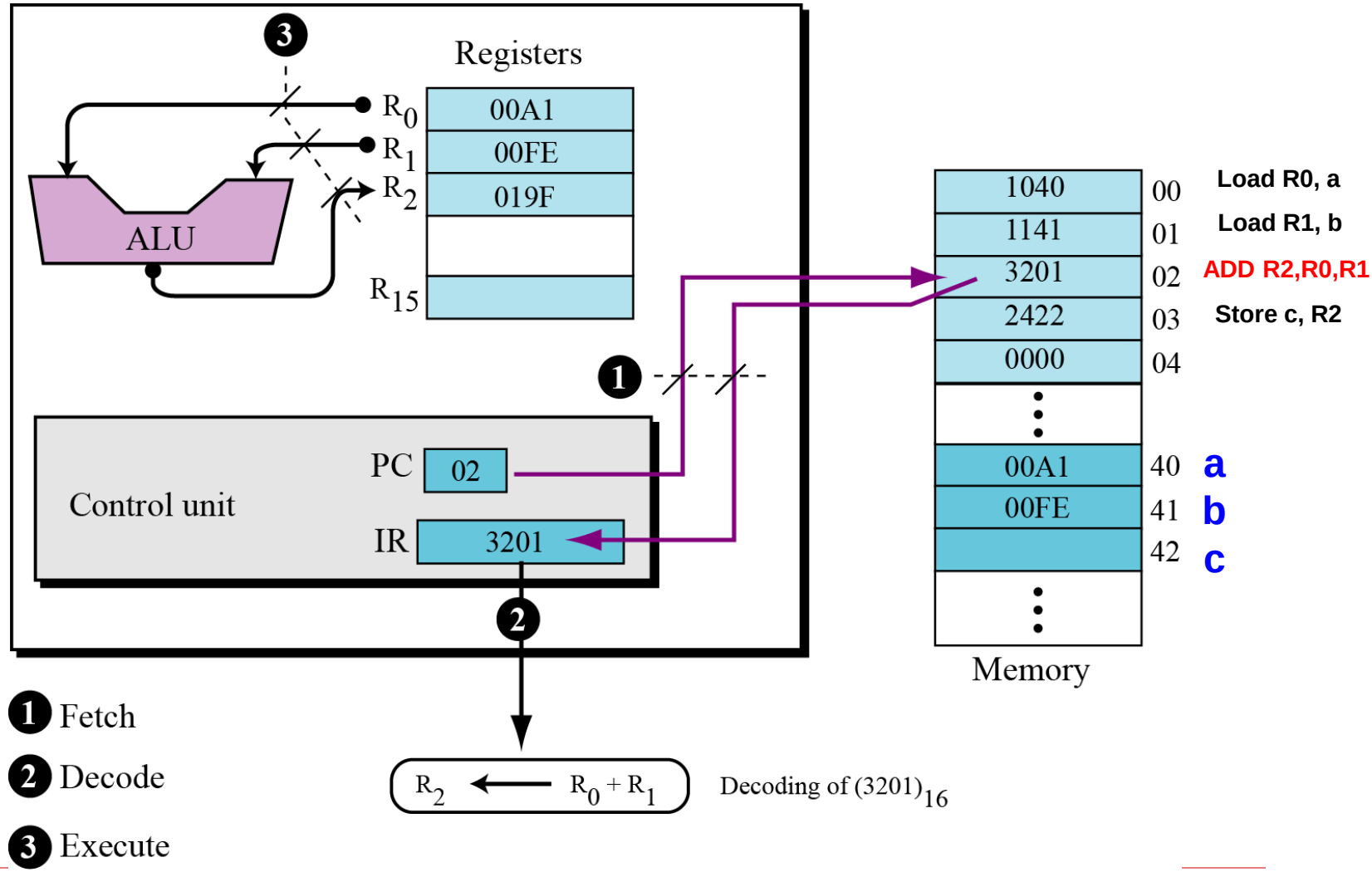
Example: c = a+ b (first step)



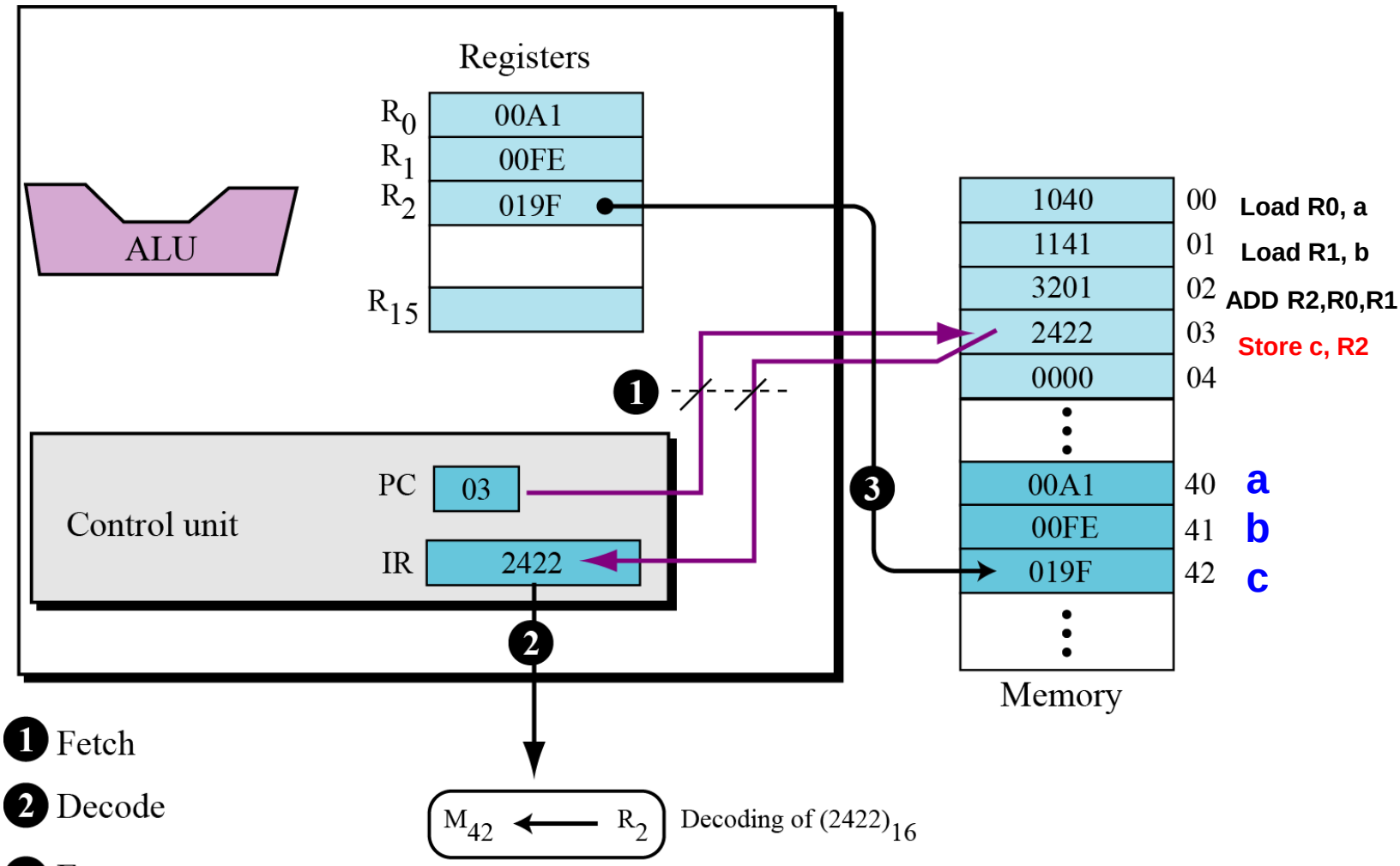
Example: c = a+ b (second step)



Example: c = a+ b (third step)

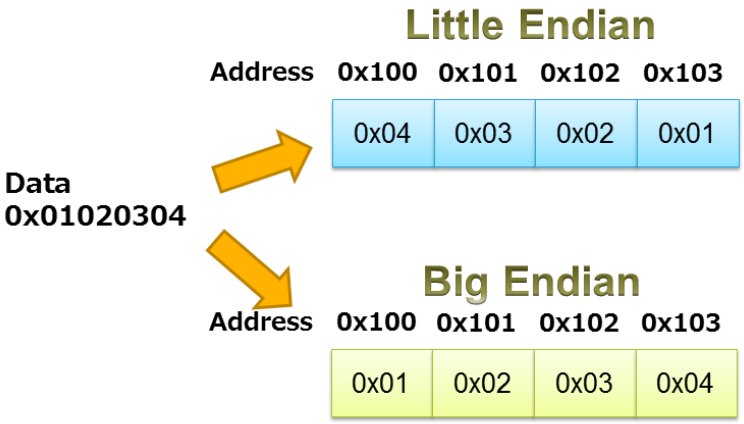


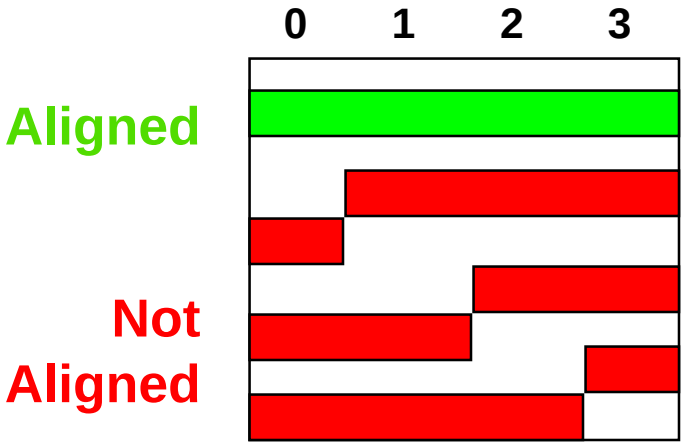
Example: c = a+ b (fourth step)



Addressing Objects: Endianess and Alignment

- **Big Endian:** address of most significant byte = word address
  - IBM 360/370, Motorola 68k, MIPS, Sparc, HP PA
- **Little Endian:** address of least significant byte = word address
  - Intel 80x86, DEC Vax, DEC Alpha (Windows NT)





Alignment: require that objects fall on address that is multiple of their size.

روشهای آدرس‌دهی

دستورات یک کامپیوتر عملی را بر روی داده ذخیره شده در حافظه و یا رجیسترهای CPUانجام میدهند. روش مشخص کردن عملوند یک دستورالعمل حالات آدرس دهی و یا addressing mode نامیده میشود.

اصولا حالات مختلف آدرس‌دهی عملوند دستور، تسهیلات زیر را در سیستم فراهم می‌آورد:

۱. قابلیت ایجاد شمارنده برای برنامه حلقه، و شاخص‌بندی در داده‌ها و هم‌چنین ایجاد اشاره‌گر حافظه و جابجایی برای کاربر فراهم می‌شود.

۲. امکان تقلیل تعداد بیت‌های قسمت آدرس دستور، فراهم می‌شود.

انواع حالت های آدرس دهی

- آدرس دهی ضمنی (Implied)
- آدرس دهی بلادرنگ (Immediate)
- آدرس دهی رجسیتری(Register)
- آدرس دهی غیر مستقیم بکمک رجیستر(Register Indirect)
- آدرس دهی مستقیم (direct)
- آدرس دهی غیر مستقیم (Indirect)
- آدرس دهی نسبی (relative)
- آدرس دهی شاخص (index)
- آدرس دهی با رجیستر پایه

آدرس دهی ضمنی

- Implied Addressing Mode

در این روش اپراند بصورت ضمنی در داخل دستورالعمل مشخص میشود.

- مثل دستور CMA (Comlement AC) که محتوی آکومولاتور را متمم میکند.
- دستورات صفر آدرسی مورد استفاده در کامپیوترهای stack machine نیز از آدرس دهی ضمنی استفاده میکنند زیرا عملوندها بطور ضمنی در بالای پشته در نظر گرفته میشوند.

آدرس دهی بلادرنگ

- Immediate Addressing Mode

در این روش مقدار عملوند در داخل خود دستورالعمل داده میشود.

این مد آدرس دهی برای مقدار دهی رجیسترها بکار میرود.

- مثل دستورات زیر:

MOV CX, **1024**

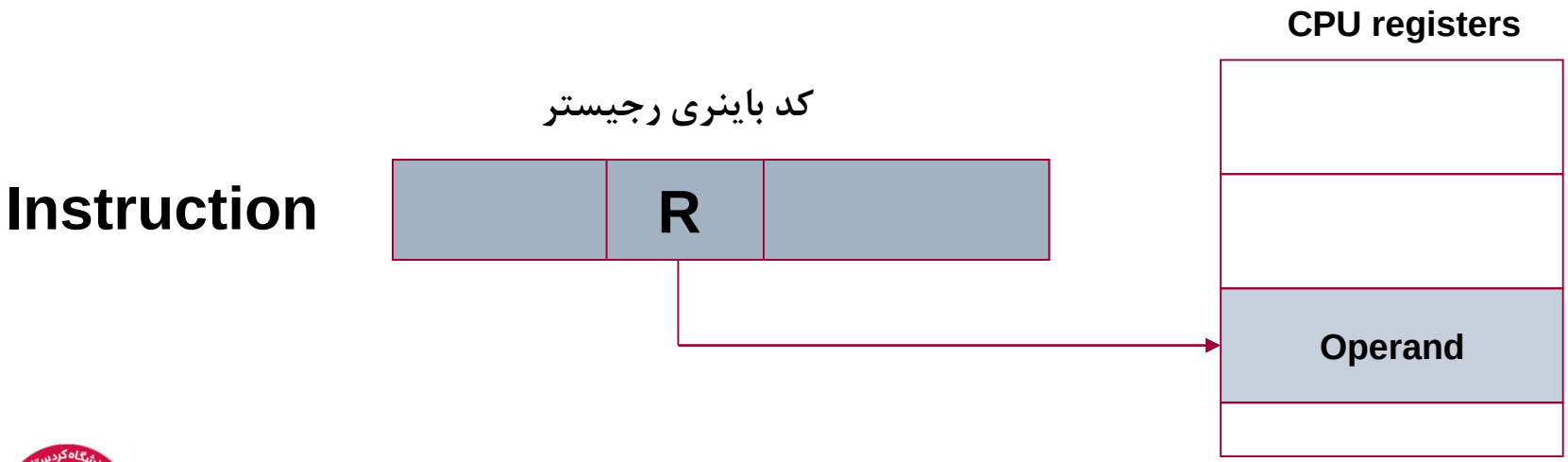
ADD R1, **20**

آدرس دهی رجیستری

- Register Addressing

در این روش عملوندها در داخل رجیسترهای پردازنده قرار دارند. با استفاده از K بیت میتوان تعداد 2^k رجیستر را مشخص نمود.

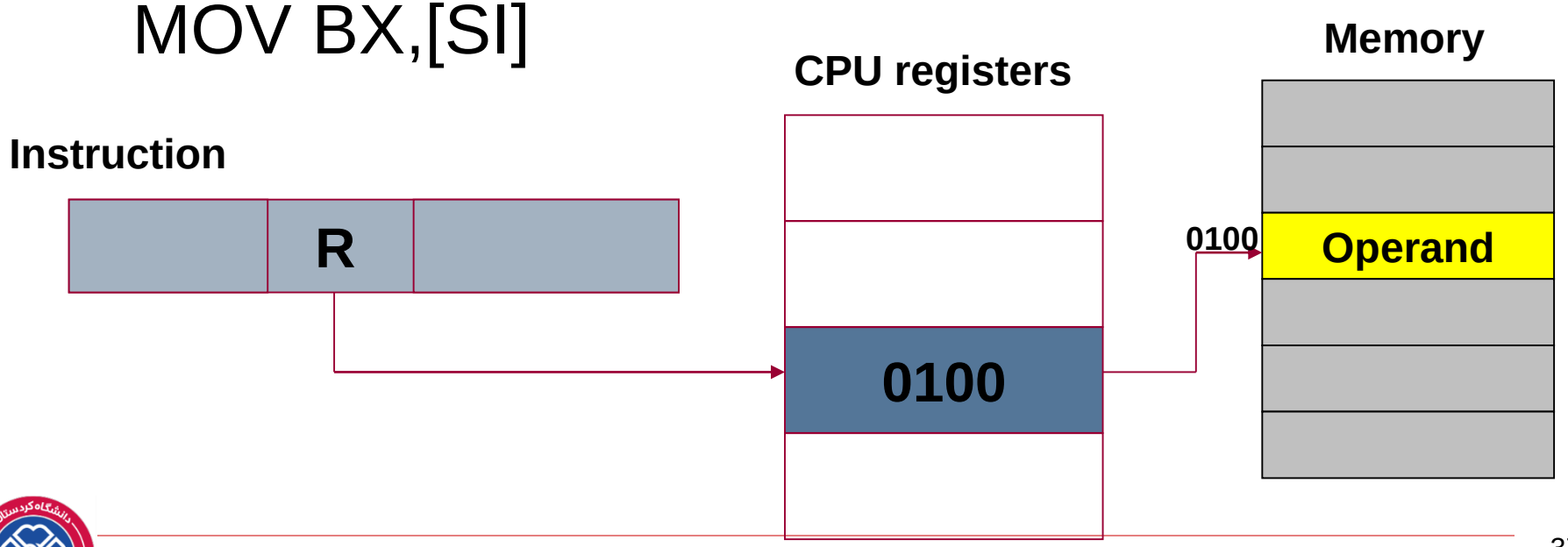
- مثل این دستور: ADD R1, R3



آدرس دهی رجیستری غیر مستقیم

- Register Indirect Addressing
- در این روش دستورالعمل رجیستری را مشخص میکند که محتوی آن آدرس عملوند در حافظه را مشخص خواهد نمود.
- مثل دستور زیر در پردازنده x86

MOV BX,[SI]



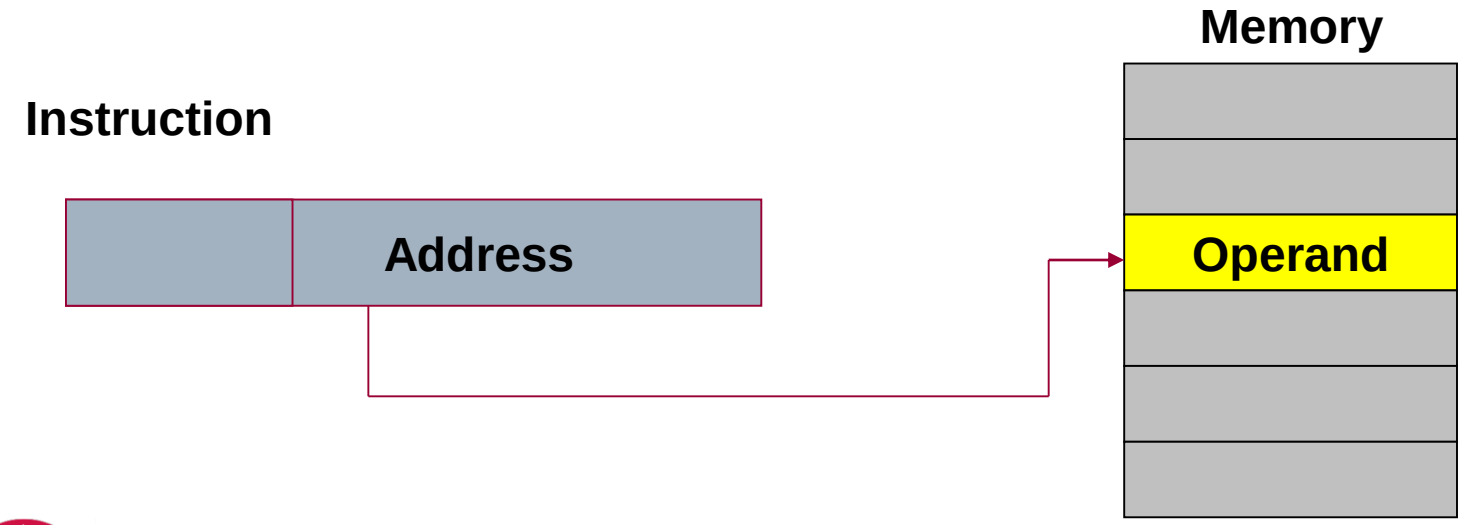
آدرس دهی مستقیم

- Direct Addressing Mode

در این روش آدرس عملوند در داخل دستورالعمل ذکر میشود.

- مثل دستور زیر در پردازنده x86

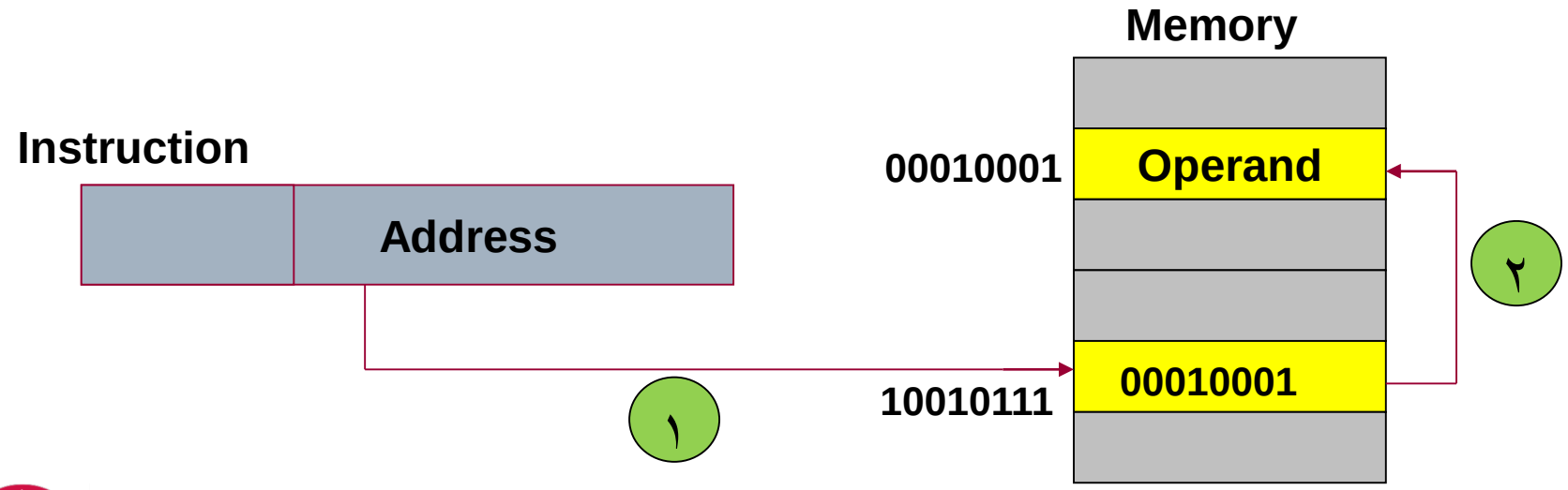
MOV AX,[3000]



آدرس دهی غیر مستقیم

- Indirect Addressing Mode

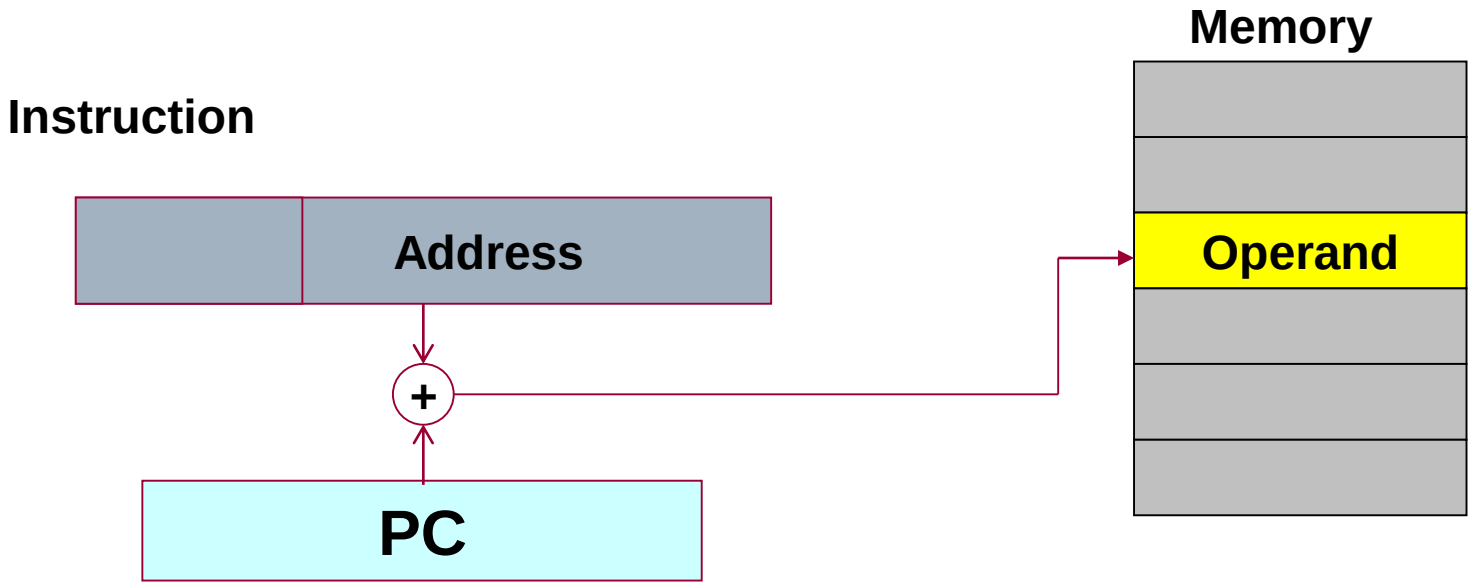
در این روش آدرس موجود در دستورالعمل محلی از حافظه را مشخص میکند که آدرس عملوند درآنجا قرار دارد. در این حالت برای دسترسی به عملوند **دوبار** رجوع به حافظه مورد نیاز است: یکبار برای پیدا کردن آدرس آن و بار دیگر برای خواندن مقدار آن.



آدرس دهی نسبی

- Relative Addressing Mode
- در این روش آدرس موثر از جمع آدرس مشخص شده در داخل دستورالعمل و محتوای PC حاصل میشود:

Effective Address= address part of instruction + content of PC

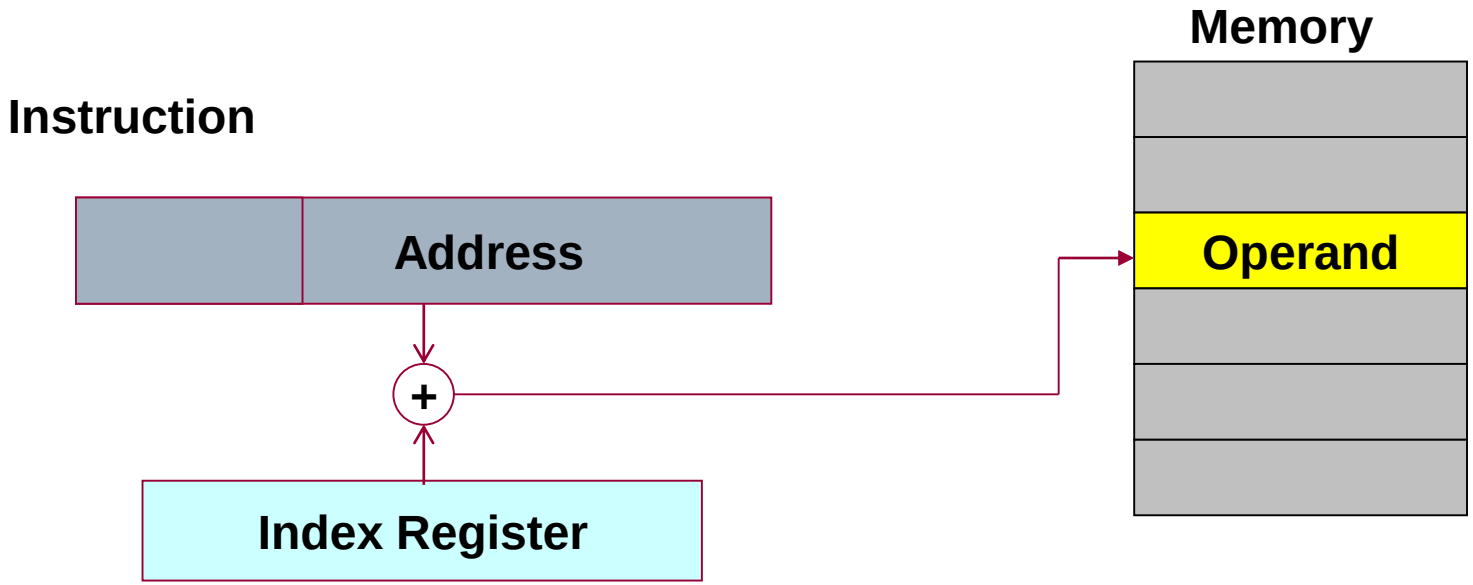


آدرس دهی شاخص دار

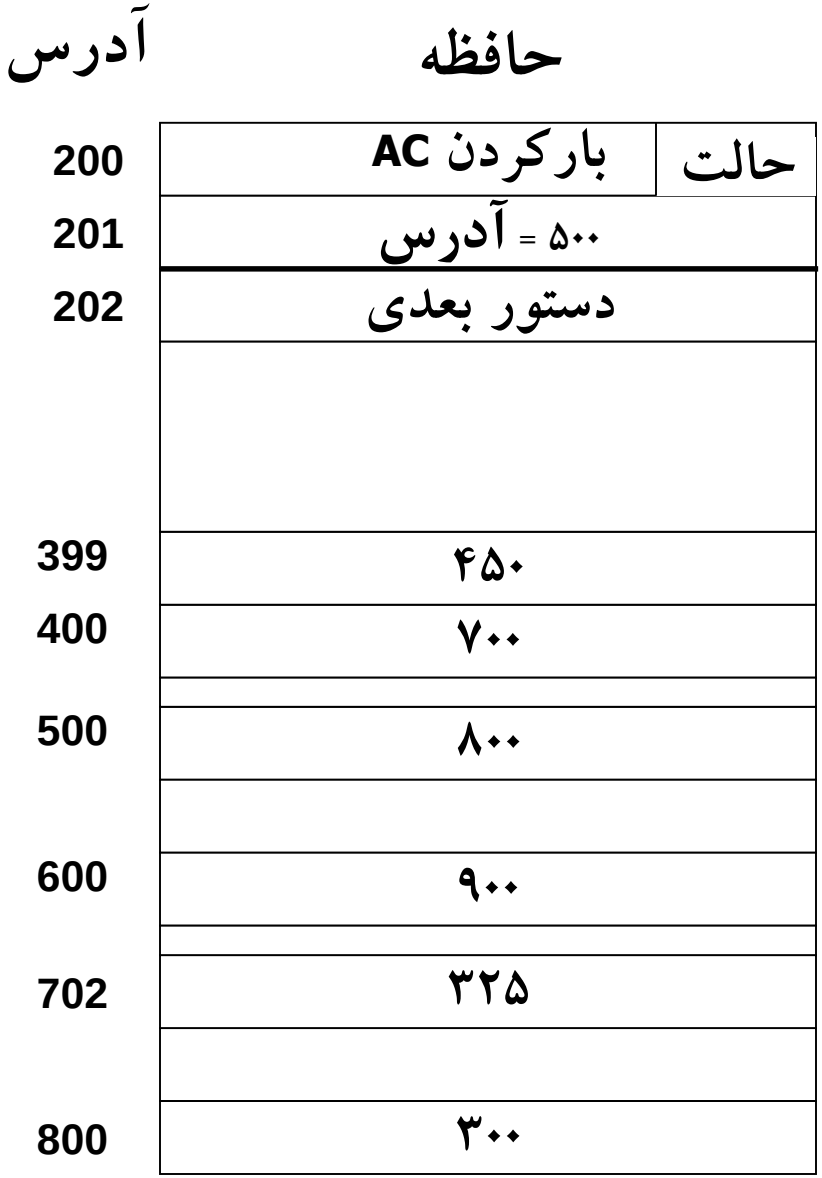
- **Index Addressing Mode**
- در این روش آدرس موثر از جمع آدرس مشخص شده در داخل دستورالعمل و محتوی یک رجیستر مخصوص که رجیستر ایندکس نامیده میشود حاصل میشود:

Effective Add.= address part of instruction + content of index register

- معمولا از این روش برای دسترسی به داده های یک آرایه استفاده میشود که محل شروع داده ها در حافظه در دستورالعمل مشخص میشود و فاصله داده مورد نظر تا محل شروع توسط رجیستر ایندکس تعیین میگردد.



مثال



مقدار آکومولاتور در صورت اجرای دستور موجود در آدرس 200 برای حالتهای مختلف آدرس دهی چیست؟

PC = 202

R1 = 400

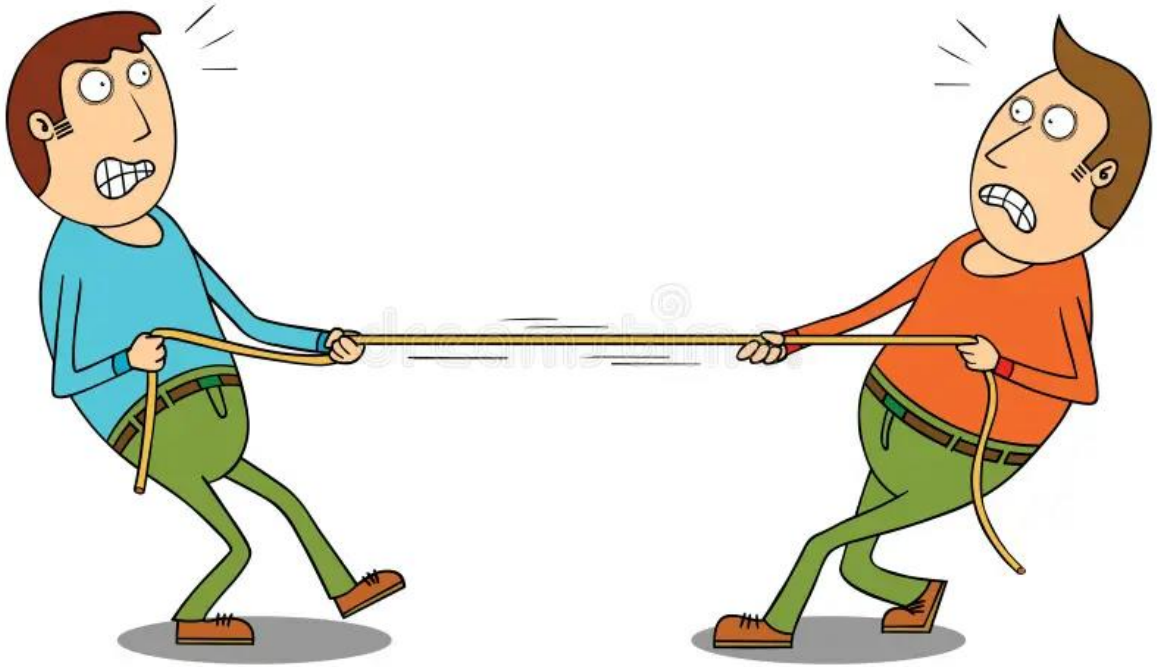
XR = 100

AC

مثال

حالات آدرس‌دهی	آدرس مؤثر	محتوای اکومولاتور
مستقیم	۵۰۰	۸۰۰
بلافصل	۲۰۱	۵۰۰
غیرمستقیم	۸۰۰	۳۰۰
نسبی	۷۰۲	۳۲۵
شاخص‌دار	۶۰۰	۹۰۰
رجیستری	–	۴۰۰
غیرمستقیم از طریق رجیستر	۴۰۰	۷۰۰

RISC vs. CISC



دو دیدگاه متفاوت در معماری کامپیوتر

کامپیوترهای با مجموعه دستورات پیچیدهCISC

- معماری یک کامپیوتر متاثر از مجموعه دستورات انتخاب شده برای آن است.
- در کامپیوتر های اولیه سعی بر این بود تا تعداد دستورات کم و ساده باشد تا پیاده سازی سخت افزاری آن ممکن باشد.
- با پیشرفت در زمینه سخت افزار و ارزان شدن آن تعداد دستورات کامپیوترها افزایش یافته و بر پیچیدگی آنها افزوده گردید. هدف این بود تا هر چه بیشتر نیاز های کاربران را در سخت افزار گنجانده وبا کاهش فاصله بین زبانهای سطح بالا و دستورات کامپیوتر کار ترجمه دستورات سطح بالا را ساده تر کنند. این نوع کامپیوترها صدها دستور و تعداد بسیار زیادی مد آدرس دهی داشتند. این کامپیوتر ها را Complex instruction set computer (CISC) مینامند.

ویژگی های کامپیوترهای CISC

- تعداد زیادی دستورالعمل دارند (مثلا 200 عدد)
- دستوراتی برای انجام کارهای ویژه دارند که معمولا بندرت مورد استفاده قرار میگیرند
- تعداد زیادی مد آدرس دهی دارند
- طول دستورالعمل ها متفاوت است
- دستوراتی دارند که عملیاتی را بر روی اپراندهای موجود در حافظه انجام میدهند
- برای اجرای دستورات به چندین کلاک نیاز هست

کامپیوترهای با مجموعه دستورات کاهش یافته RISC

- معماری RISC ابداعی مهم در زمینه سازمان کامپیوتر بشمار میرود.
- در این معماری سعی شده است تا با ساده تر کردن مجموعه دستورالعمل ها بر قدرت پردازنده افزوده شود.

ویژگی های RISC

- دستورات ساده و کمی دارند
  - هدف این است که بتوان دستورات را بسرعت اجرا کرد. اغلب دستورات RISC در یک سیکل اجرا میشوند (بعد ازواکشی و رمزگشایی)
  - چون دستورات در زمان مشابهی اجرا میشوند عمل pipeline دارای بازده بالایی خواهد بود.
  - کم بودن دستورات باعث ساده شدن واحد کنترل ومسیرهای ارتباطی میشود که منجر به کم شدن تعداد ترانزیستورهای مورد نیاز برای ساخت پردازنده میشود اینکار سرعت پردازنده را نیز افزایش میدهد.
  - در ضمن واحد کنترل بروش سیم بندی ساخته می شود.

ویژگی های RISC

- **دسترسی به حافظه محدود به دستورات load و Store است**
  - **از آنجائیکه عمل رجوع به حافظه معمولا در یک سیکل امکانپذیر نمیباشد تمهیداتی در کامپایلر استفاده میشود تا pipelineبطور موثرتری استفاده شود.**
  - **باقی دستورات بر روی محتوی رجیسترها عمل میکنند.**

ویژگی های RISC

- تعداد مدهای آدرس دهی آنها کم است

  –معمولا فقط از مدهای آدرس دهی زیر استفاده میشود:

  - register addressing
  - direct addressing
  - register indirect addressing
  - displacement addressing

ویژگی های RISC

- دستورات طول و فرمت یکسانی دارند

- اینکار باعث میشود تا خواندن دستورات و رمزگشائی آنها سریع و ساده باشد. زیرا لازم نیست تا معلوم شدن طول دستور برای رمزگشایی آن صبر کرد.

- یکسانی فرمت باعث سهولت رمزگشایی میگردد زیرا کد و ادرس همه دستورات در محل یکسانی قرار دارند.

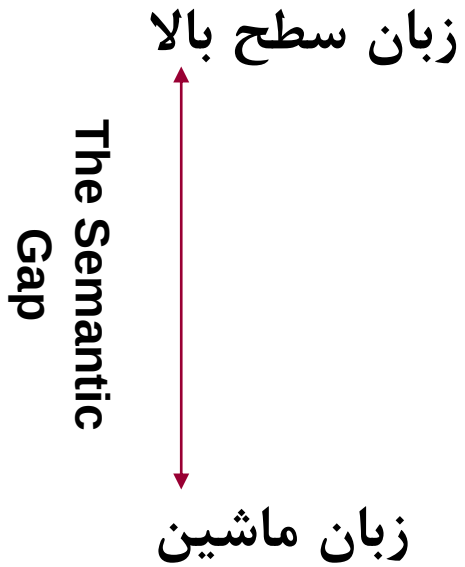
ویژگی های RISC

- تعداد رجیسترهای زیادی دارند.
  - کم شدن دستورات باعث کوچگ شدن واحد کنترل و آزاد شدن فضا برای گنجاندن تعداد بیشتری رجیستر میشود.
  - متغیرهای محلی، نتایج میانی و پارامترهای توابع در داخل رجیسترها ذخیره شده و تعداد رجوع به حافظه کاهش پیدا می کند.

سایر ویژگی های RISC

- استفاده ازتکنیک pipelining که باهمپوشانی سیکل های واکشی، رمزگشائی و اجرا شده و نهایتا به اجرای هر دستور در یک سیکل منجر میشود.
- تعداد زیادی رجیستر در داخل CPU وجود دارد
- استفاده از تکنیک همپوشانی register windows که باعث افزایش سرعت فراخوانی تابع و بازگشت از آن میشود.
- حمایت از کامپایلر برای افزایش کارائی در ترجمه برنامه های سطح بالا

The Semantic Gap



با پیشرفت زبانهای سطح بالا اختلاف بین برنامه های سطح بالا و زبان ماشین گسترش یافت. هر یک از معماریهای فوق روش مختلقی برای کاهش این اختلاف معرفی کردند تا ترجمه زبانهای سطح بالا بطرز موثرتری انجام شود.

- **نگرش CISC: دستورات زبان ماشین پیچیده تر و به زبان سطح بالا نزدیکتر گردد.**
- **نگرش RISC: دستورات زبان ماشین ساده تر و به نیازهای برنامه نزدیکتر گردد.**

مقایسه RISC, CISC

- برای مثال ترجمه دستور a = a * b برای دو معماری فوق بصورت زیر خواهد بود:

CISC	MULT a, b
RISC	LOAD R1, a LOAD R2, b MUL R1,R2 STORE a, R1

- **برای مثال ترجمه دستور a = a * b برای دو معماری فوق بصورت زیر خواهد بود:**
- **در معماری CISC سعی بر این است که با تعداد کمی دستورزبان ماشین عمل مورد نظر انجام شود. در حالیکه درRISC سعی بر این است که از دستوراتی استفاده شود که هر دستور در یک پالس قابل اجرا باشند. در نتیجه برنامه RISC دارای چندین خط دستور خواهد بود که احتیاج به حافظه بیشتری دارد.**

مقایسه RISC, CISC

- فرمول محاسبه کارائی

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

CISC سعی میکند تا این قسمت کاهش پیدا کند

RISC سعی میکند تا این قسمت کاهش پیدا کند

پیشگامان RISC

- ایده RISC در دهه 80 معرفی گردید. از جمله پیشگامان آن میتوان به موارد زیر اشاره نمود

  - IBM 801
  - Stanford MIPS
  - Berkeley RISC 1 and 2

کاربردهای RISC

تعدادی از پردازنده های مشهور مبتنی بر RISC عبارتند از:

- ARM/StrongARM که در تلفنهای همراه استفاده میشود
- MIPS Rxx00 که در برخی دستگاه های بازی مورد استفاده است
- Hitachi SH مورد استفاده در سیستم های embedded
- POWER/PowerPC مورد استفاده در کامپیوترهایMAC

مقایسه اجمالی RISC و CISC

CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy


Questions