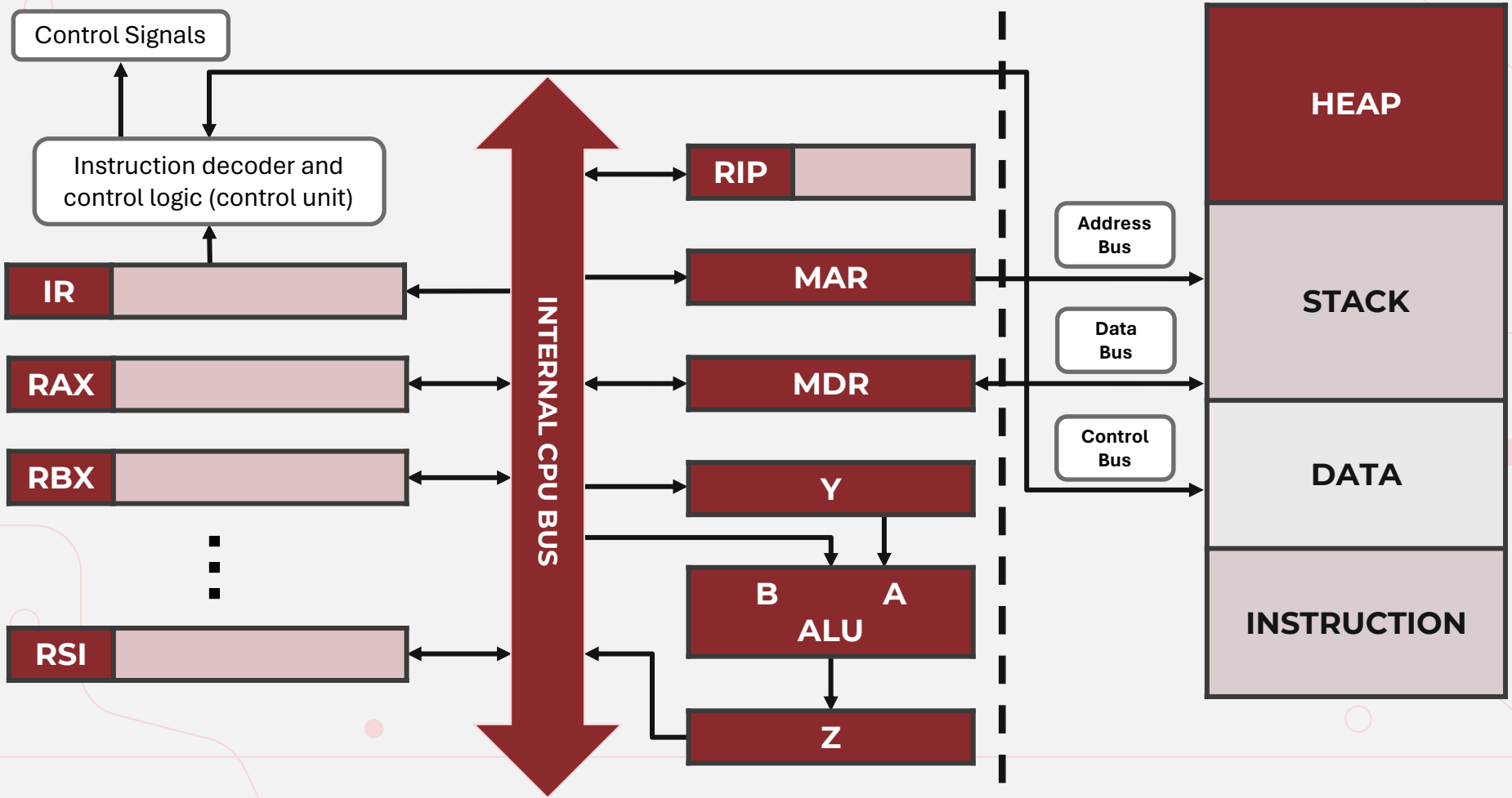
Assembly Language Lecture Series:

Software Architecture: x86-64 Memory

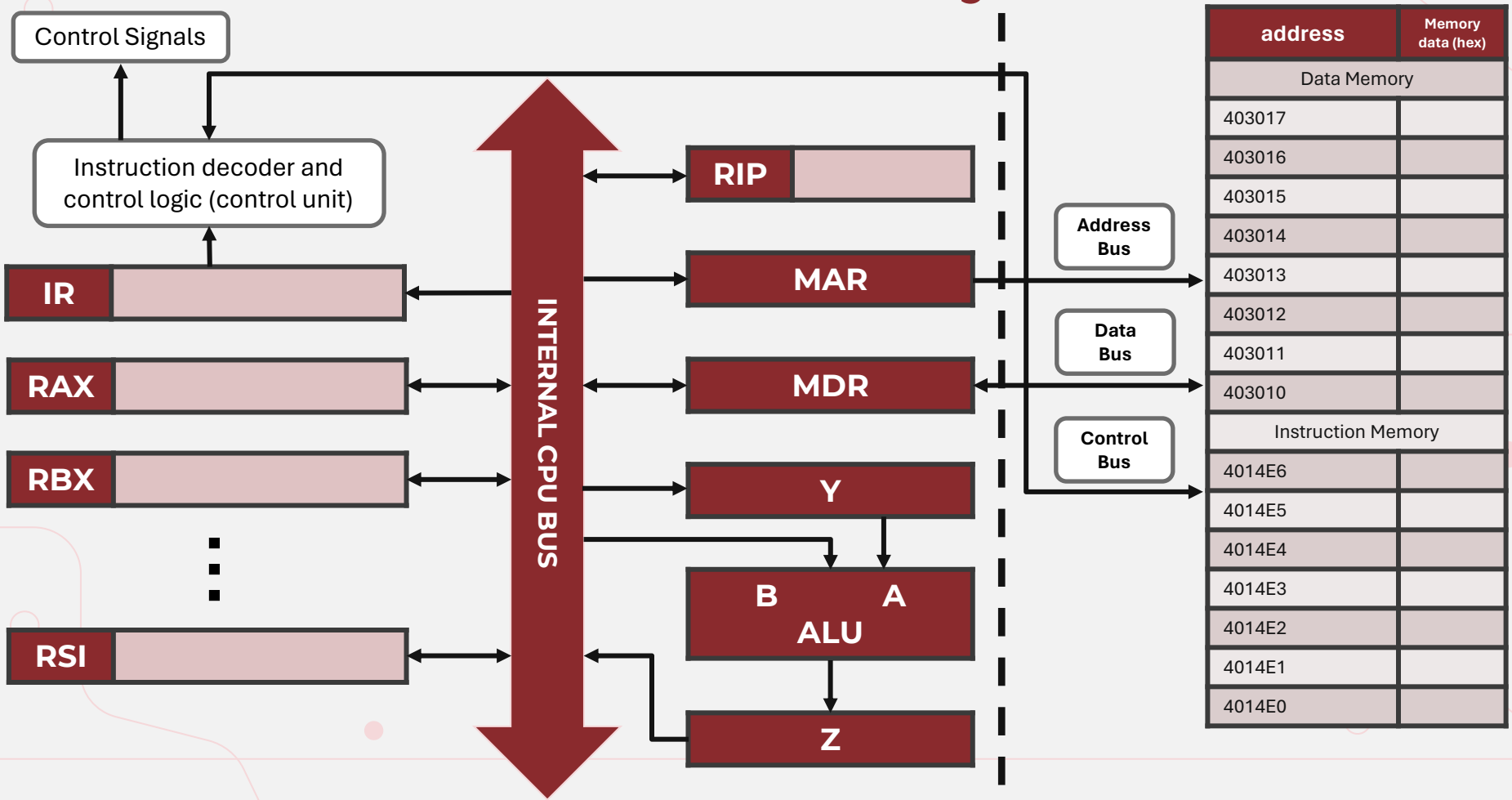
Sensei RL Uy, College of Computer Studies,
De La Salle University, Manila, Philippines

Copyright Notice

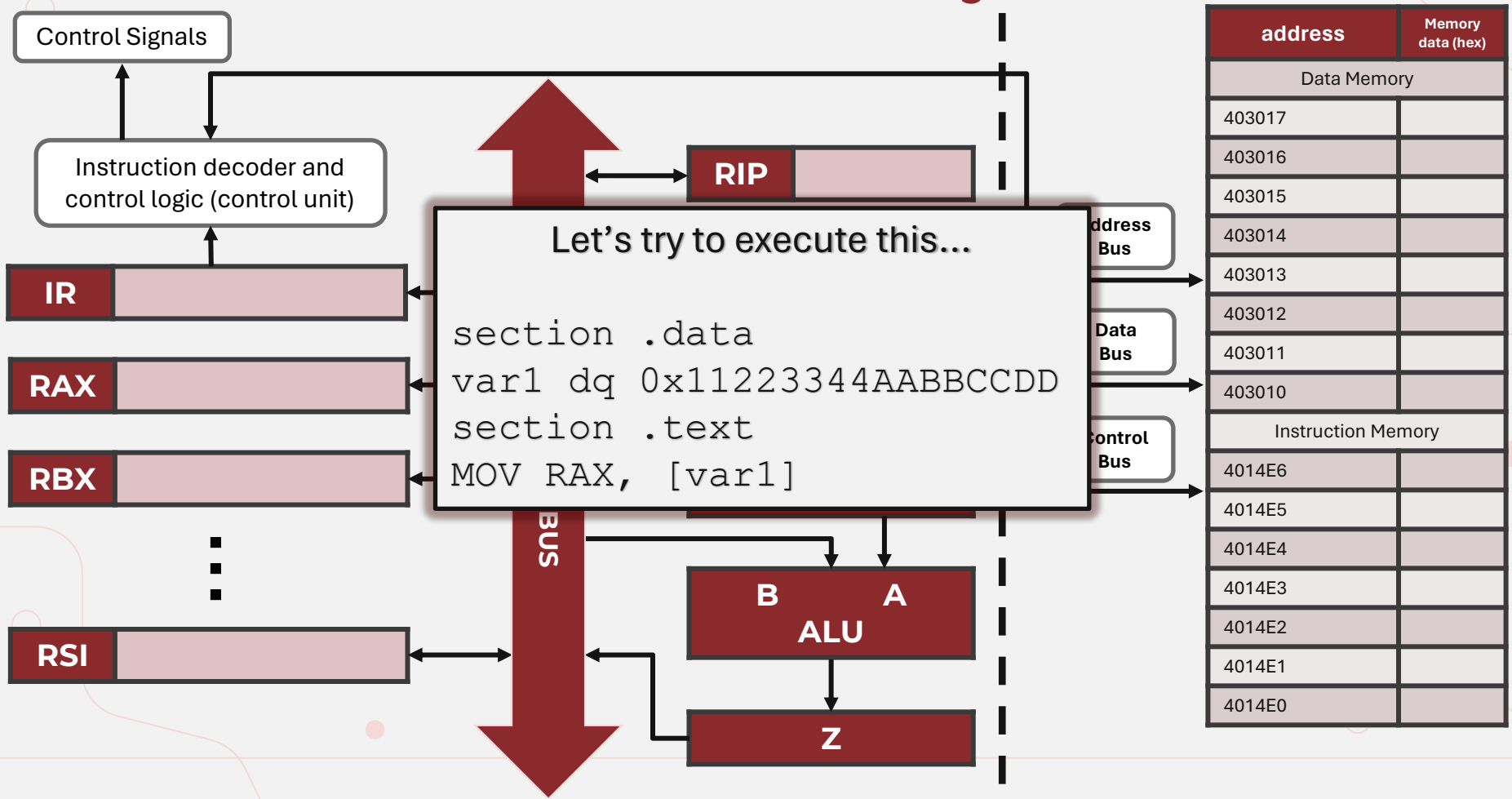
Software Architecture



Software Architecture: Memory



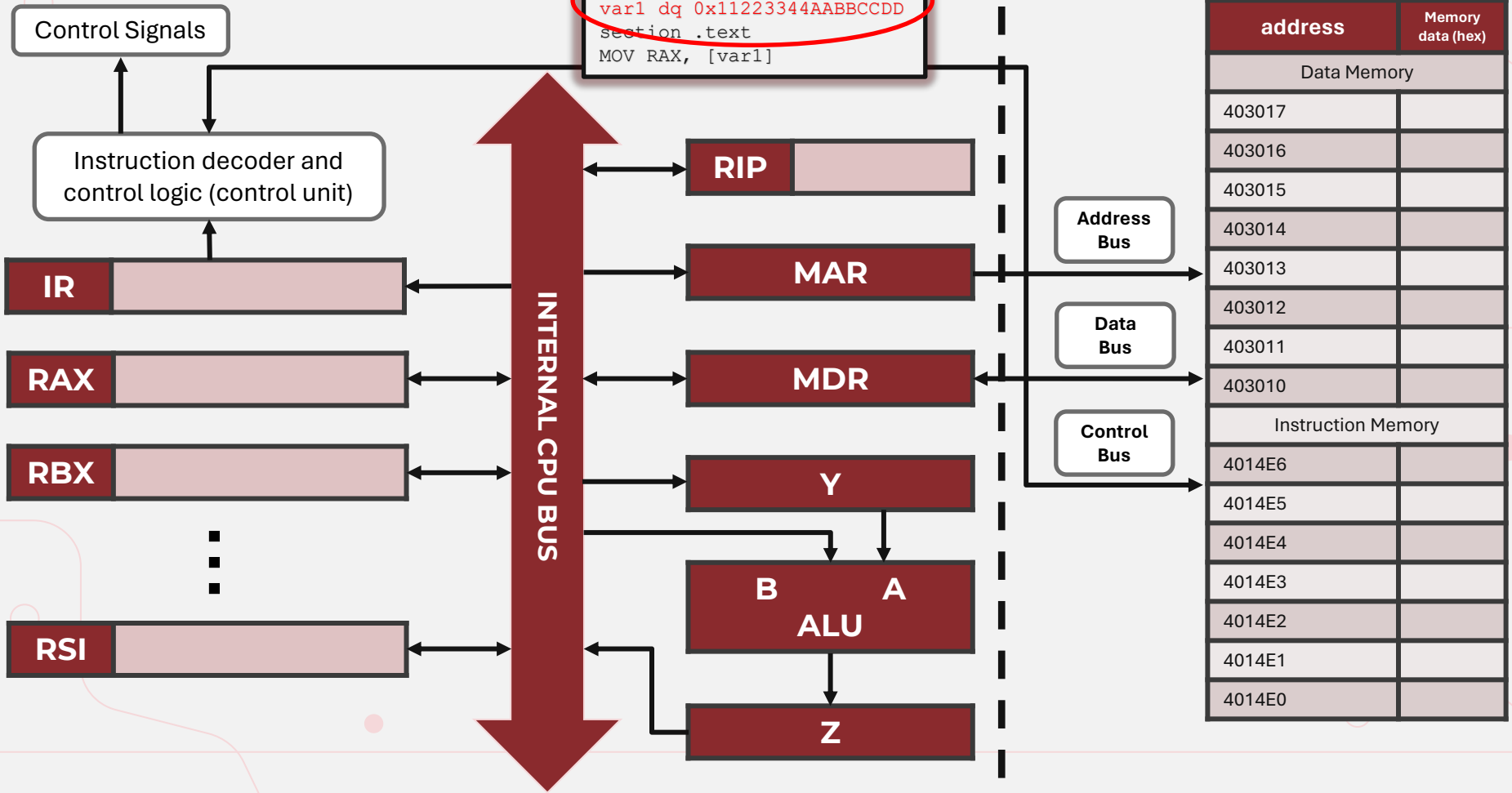
Software Architecture: Memory



Software Architect ry

Let's try to execute this...

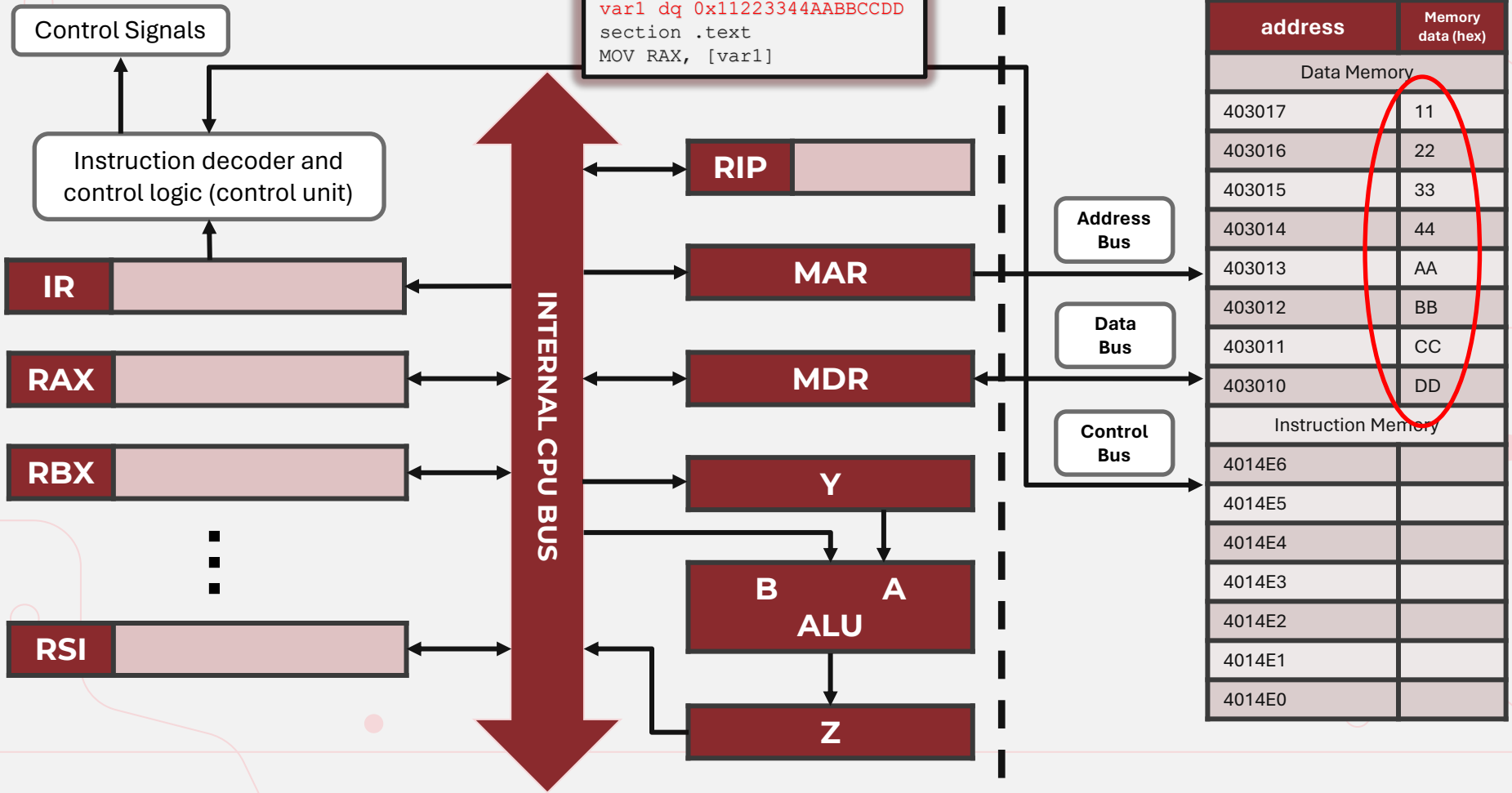
```
section .data
var1 dq 0x11223344AABBCCDD
section .text
MOV RAX, [var1]
```



Software Architect ory

Let's try to execute this...

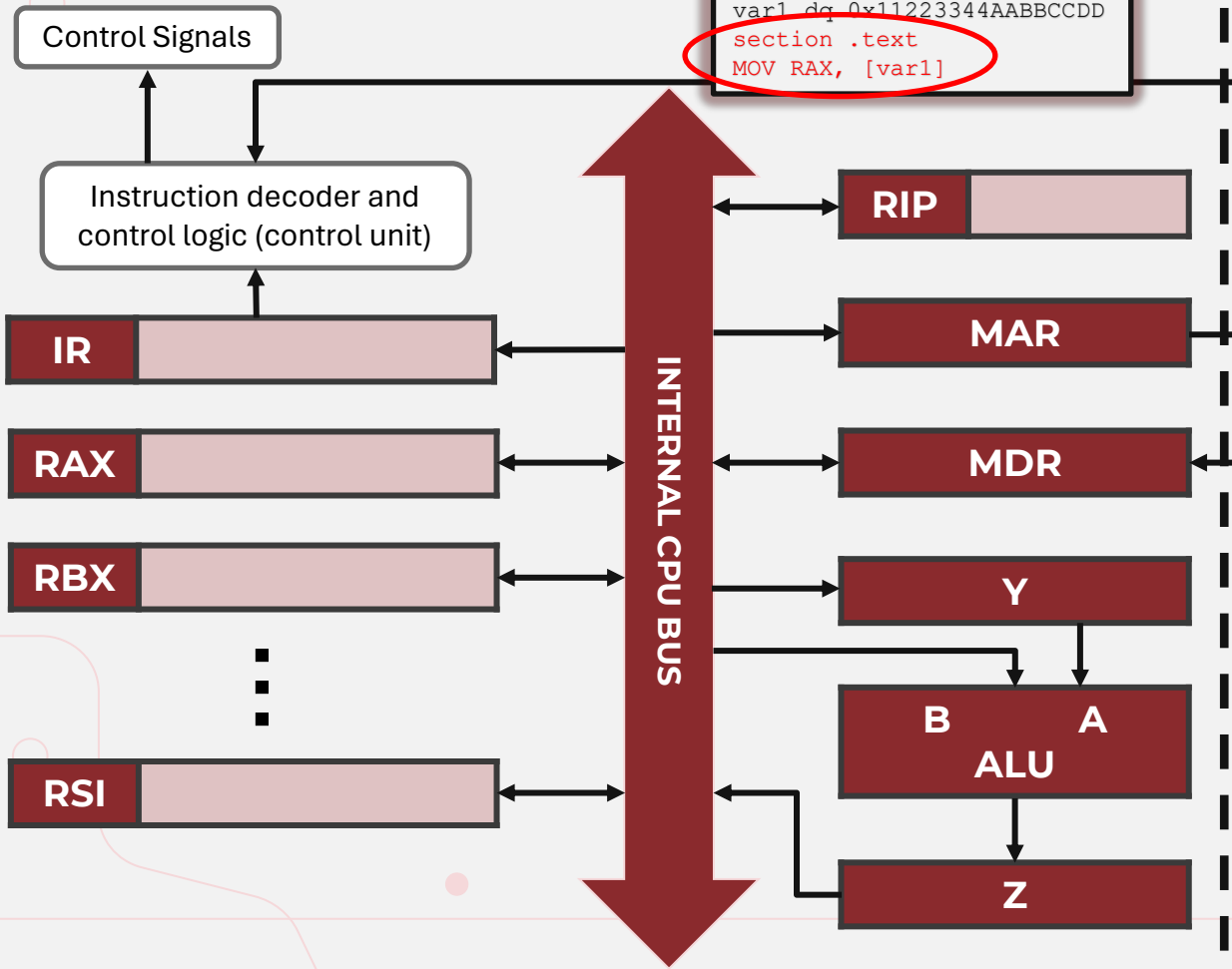
```
section .data
var1 dq 0x11223344AABBCCDD
section .text
MOV RAX, [var1]
```



Software Architect ry

Let's try to execute this...

```
section .data
var1 dq 0x11223344AABBCCDD
section .text
MOV RAX, [var1]
```

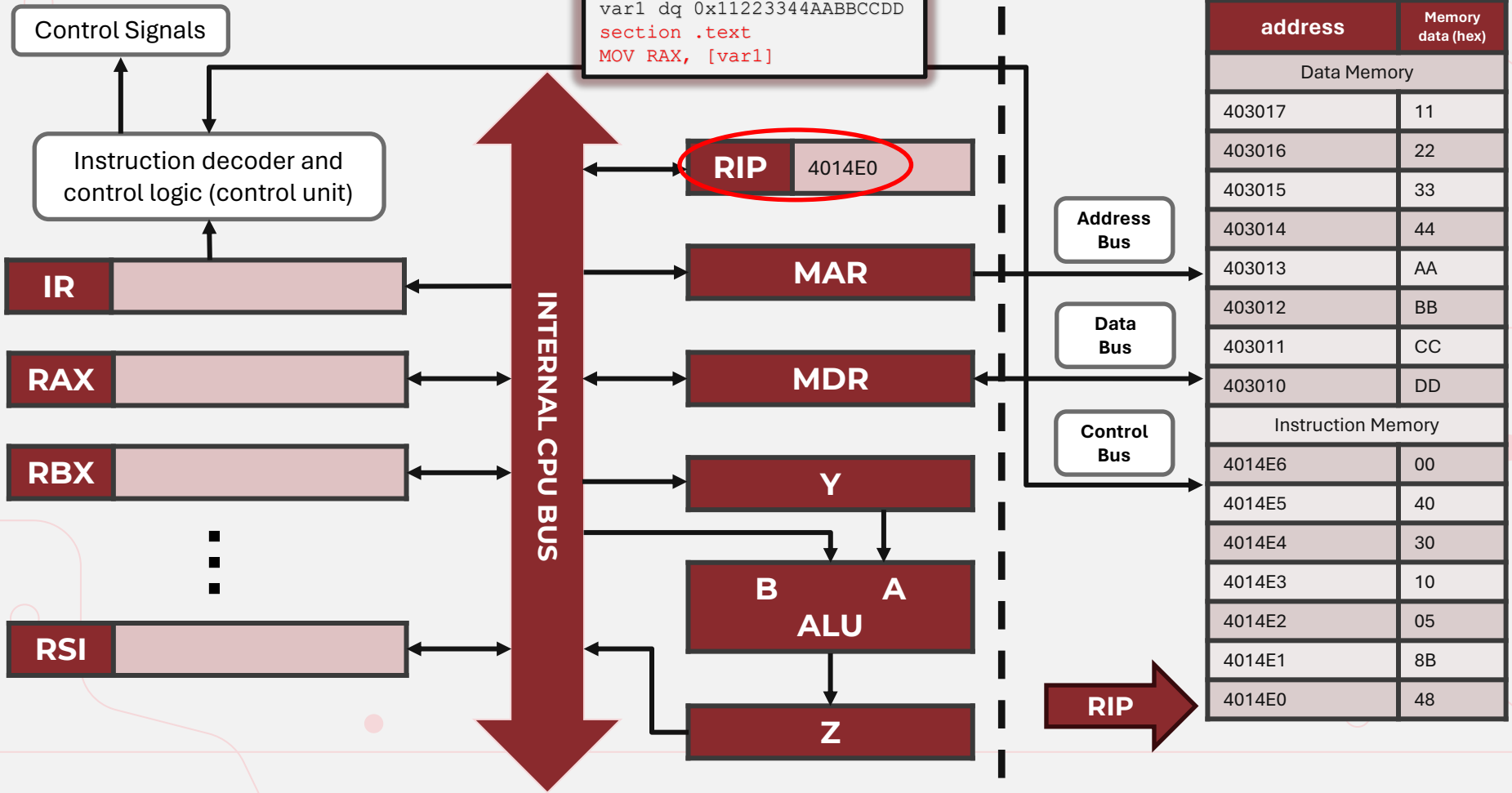


address	Memory data (hex)
Data Memory	
403017	11
403016	22
403015	33
403014	44
403013	AA
403012	BB
403011	CC
403010	DD
Instruction Memory	
4014E6	00
4014E5	40
4014E4	30
4014E3	10
4014E2	05
4014E1	8B
4014E0	48

Software Architect ry

Let's try to execute this...

```
section .data
var1 dq 0x11223344AABBCCDD
section .text
MOV RAX, [var1]
```

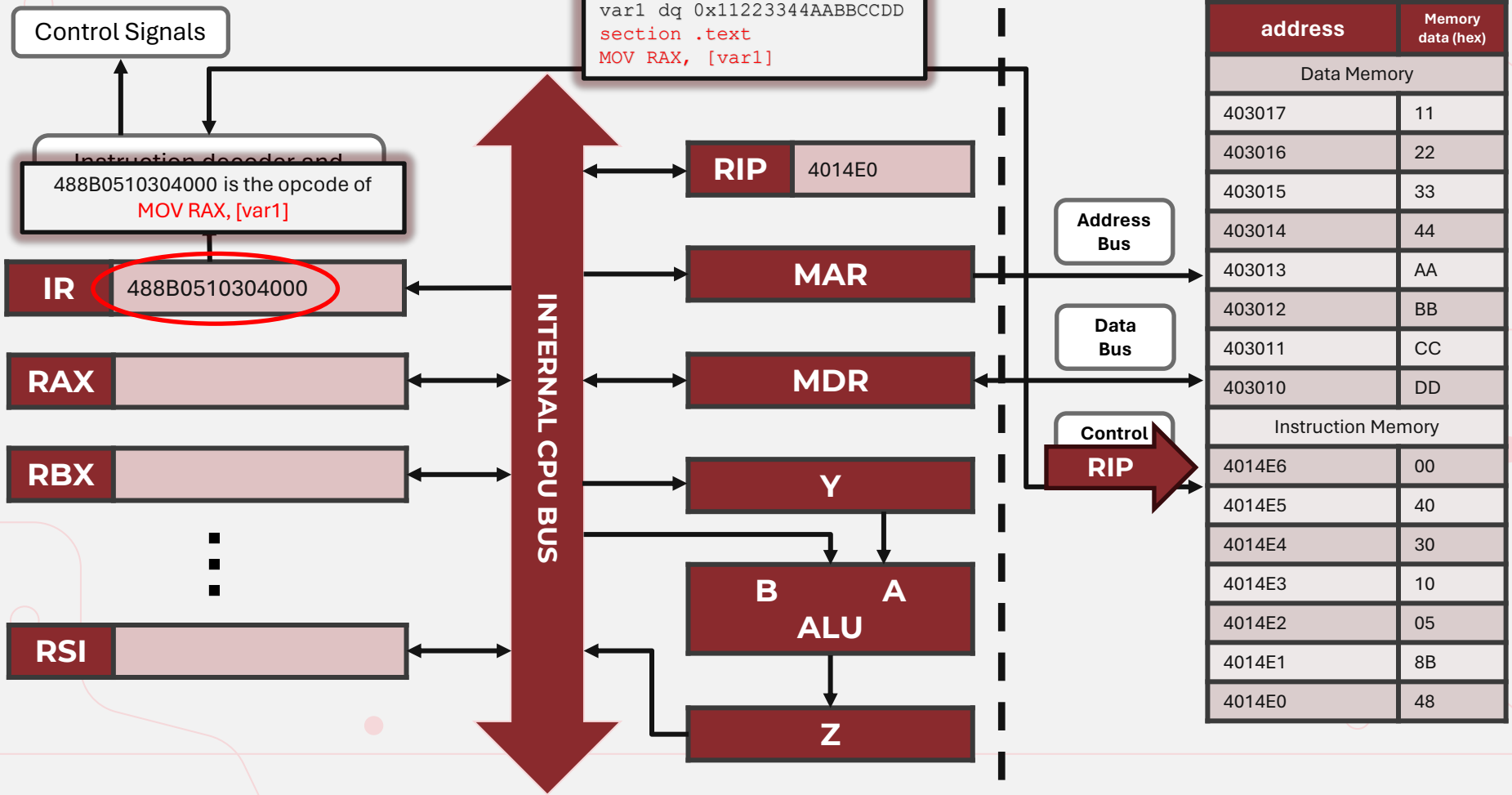


address	Memory data (hex)
Data Memory	
403017	11
403016	22
403015	33
403014	44
403013	AA
403012	BB
403011	CC
403010	DD
Instruction Memory	
4014E6	00
4014E5	40
4014E4	30
4014E3	10
4014E2	05
4014E1	8B
4014E0	48

Software Architect ry

Let's try to execute this...

```
section .data
var1 dq 0x11223344AABBCCDD
section .text
MOV RAX, [var1]
```

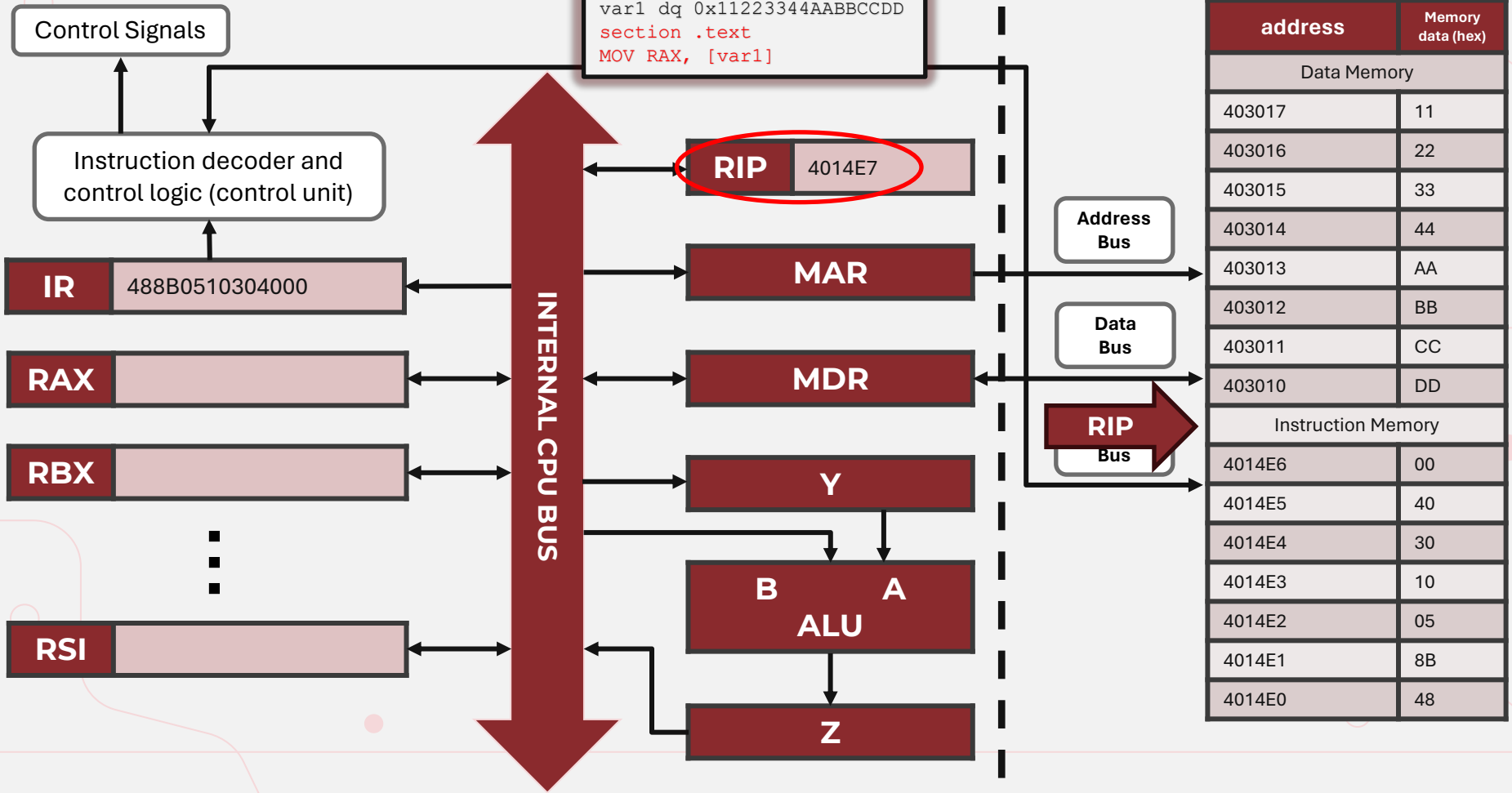


address	Memory data (hex)
Data Memory	
403017	11
403016	22
403015	33
403014	44
403013	AA
403012	BB
403011	CC
403010	DD
Instruction Memory	
4014E6	00
4014E5	40
4014E4	30
4014E3	10
4014E2	05
4014E1	8B
4014E0	48

Software Architect ory

Let's try to execute this...

```
section .data
var1 dq 0x11223344AABBCCDD
section .text
MOV RAX, [var1]
```

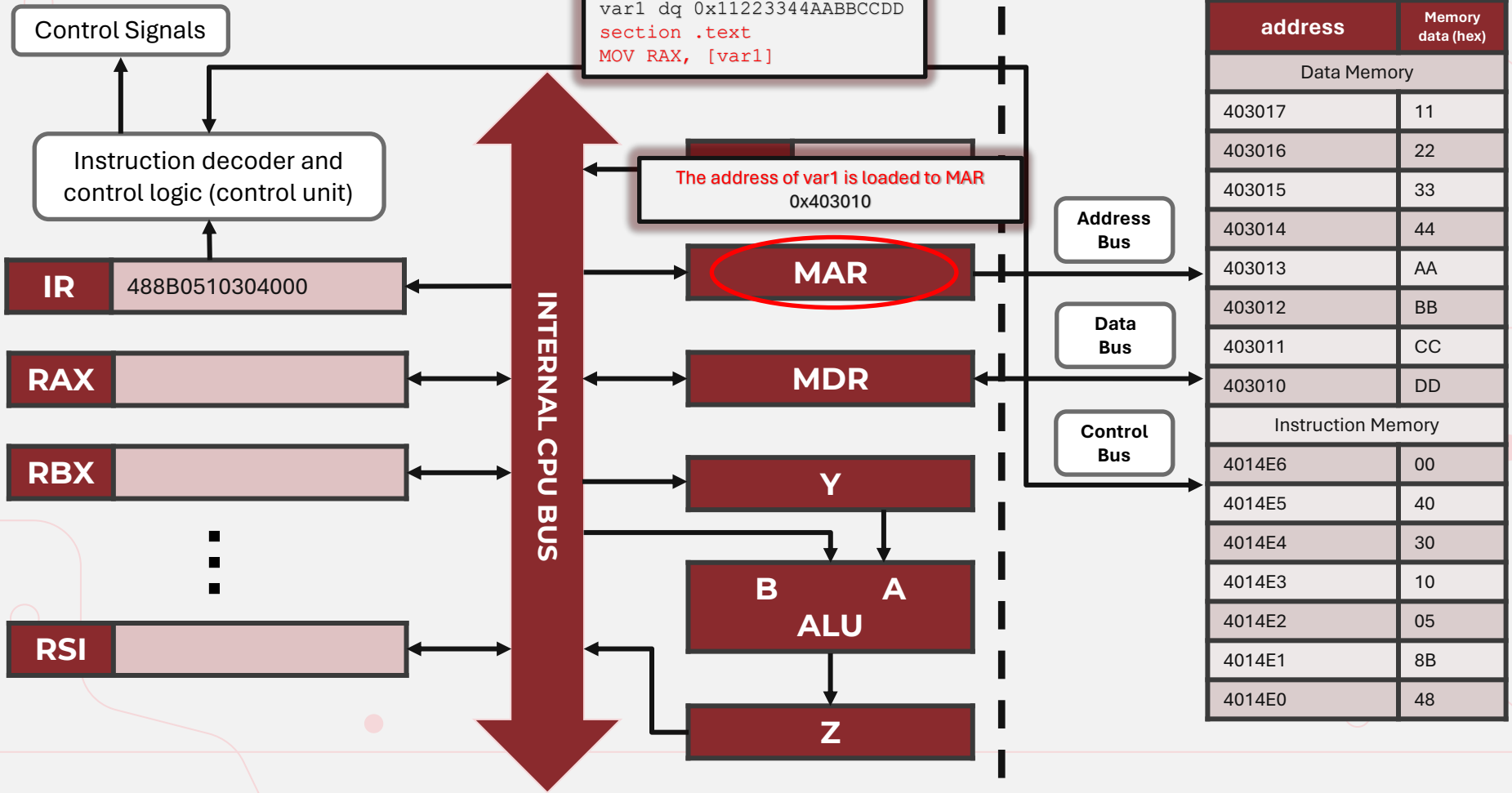


address	Memory data (hex)
Data Memory	
403017	11
403016	22
403015	33
403014	44
403013	AA
403012	BB
403011	CC
403010	DD
Instruction Memory	
4014E6	00
4014E5	40
4014E4	30
4014E3	10
4014E2	05
4014E1	8B
4014E0	48

Software Architect ry

Let's try to execute this...

```
section .data
var1 dq 0x11223344AABBCCDD
section .text
MOV RAX, [var1]
```

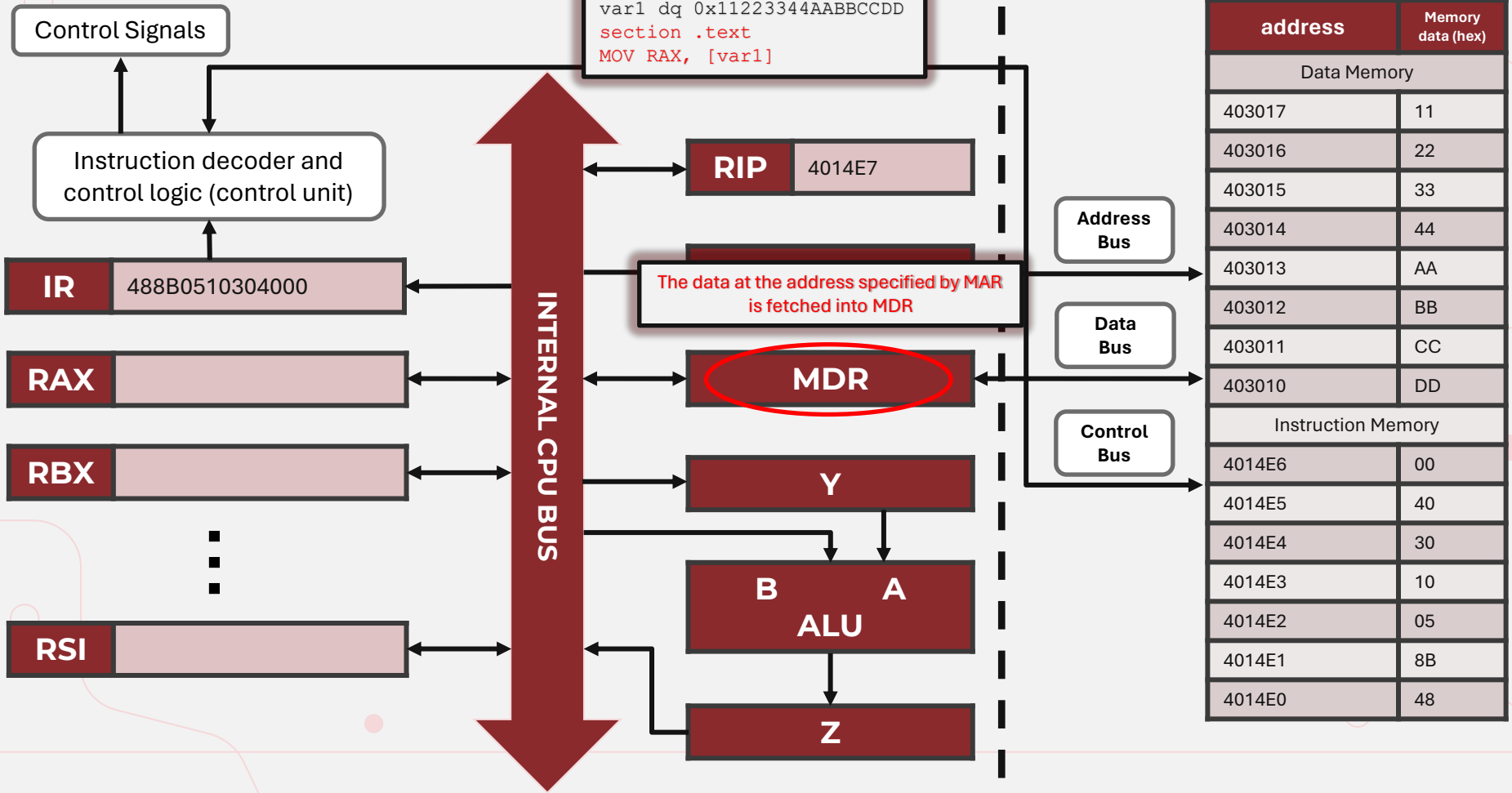


address	Memory data (hex)
Data Memory	
403017	11
403016	22
403015	33
403014	44
403013	AA
403012	BB
403011	CC
403010	DD
Instruction Memory	
4014E6	00
4014E5	40
4014E4	30
4014E3	10
4014E2	05
4014E1	8B
4014E0	48

Software Architect ry

Let's try to execute this...

```
section .data
var1 dq 0x11223344AABBCCDD
section .text
MOV RAX, [var1]
```

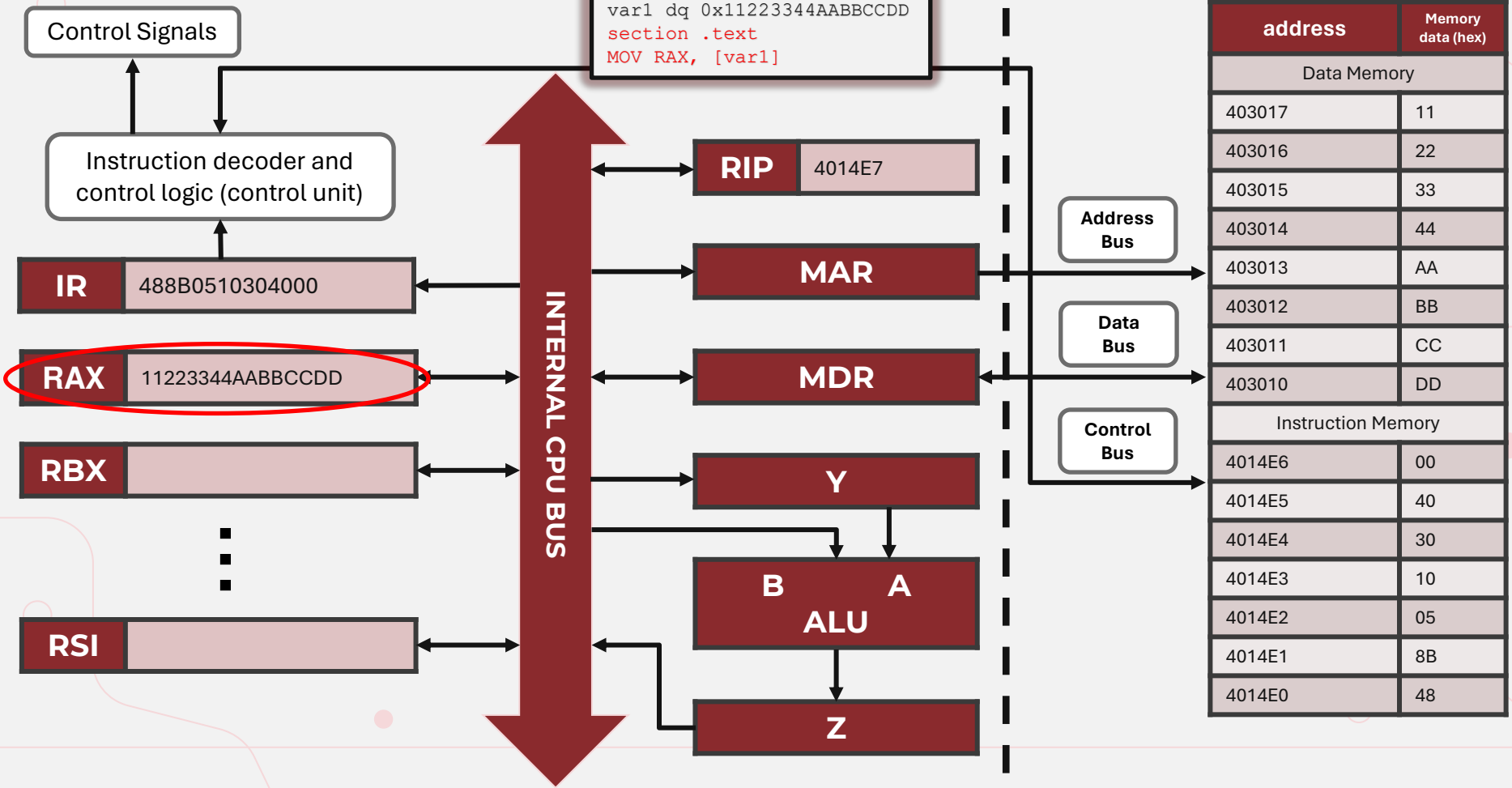


address	Memory data (hex)
Data Memory	
403017	11
403016	22
403015	33
403014	44
403013	AA
403012	BB
403011	CC
403010	DD
Instruction Memory	
4014E6	00
4014E5	40
4014E4	30
4014E3	10
4014E2	05
4014E1	8B
4014E0	48

Software Architect ory

Let's try to execute this...

```
section .data
var1 dq 0x11223344AABBCCDD
section .text
MOV RAX, [var1]
```



address	Memory data (hex)
Data Memory	
403017	11
403016	22
403015	33
403014	44
403013	AA
403012	BB
403011	CC
403010	DD
Instruction Memory	
4014E6	00
4014E5	40
4014E4	30
4014E3	10
4014E2	05
4014E1	8B
4014E0	48

Software Architecture: Memory

- Each memory location is *n*-bit in size
  - Usually 8-bit (byte addressable)
- Accessing the memory requires distinct names [*label* or *var name*] or addresses
- 2^k address constitute the address space of the computer (i.e., from 0 to 2^k -1 successive locations in the memory)

Label / var	address (hex)	Memory data (hex)
NANCY	40301C	FF
MAMA	40301B	12
KILO	40301A	CA
JULIET	403019	FE
INDIA	403018	1B
HOTEL	403017	F0
GAMMA	403016	DE
FOXTROT	403015	BC
ECHO	403014	9A
DELTA	403013	78
CHARLIE	403012	56
BETA	403011	34
ALPHA	403010	12

Software Architecture: Memory

- Data can be multi-byte (16-bit *word;* 32-bit *doubleword;* 64-bit *quadword;* 128-bit *octaword*)
- **Endianness** refers to the convention used to interpret the byte ordering of multi-byte stored in the computer memory
  - 2 types: **Big Endian** and **Little Endian**

Label / var	address (hex)	Memory data (hex)
NANCY	40301C	FF
MAMA	40301B	12
KILO	40301A	CA
JULIET	403019	FE
INDIA	403018	1B
HOTEL	403017	F0
GAMMA	403016	DE
FOXTROT	403015	BC
ECHO	403014	9A
DELTA	403013	78
CHARLIE	403012	56
BETA	403011	34
ALPHA	403010	12

Little Endian

The address of a datum is the address of the least-significant byte (LSB)

- Example: What is a 16-bit (2-byte data) found in address 403012?
  - 0x7856
- Example: What is a 32-bit (4-byte data) found in address 403018?
  - 0x12CAFE1B

address (hex)	Memory data (hex)
40301C	FF
40301B	12
40301A	CA
403019	FE
403018	1B
403017	F0
403016	DE
403015	BC
403014	9A
403013	78
403012	56
403011	34
403010	12

Big Endian

The address of a datum is the address of the most-significant byte (MSB)

- Example: What is a 16-bit (2-byte data) found in address 403012?
  - 0x5678
- Example: What is a 32-bit (4-byte data) found in address 403018?
  - 0x1BFECA12

address (hex)	Memory data (hex)
40301C	FF
40301B	12
40301A	CA
403019	FE
403018	1B
403017	F0
403016	DE
403015	BC
403014	9A
403013	78
403012	56
403011	34
403010	12

Memory Alignment

- Memory alignment: access to memory larger than a byte must be aligned
- Problems with misaligned memory: requires multiple aligned memory references
- **Aligned if:** data of size s bytes at byte address A is A mod s == 0

address (hex)	Memory data (hex)
40301C	FF
40301B	12
40301A	CA
403019	FE
403018	1B
403017	F0
403016	DE
403015	BC
403014	9A
403013	78
403012	56
403011	34
403010	12

Memory Alignment

- For 16-bit data, the address aligned at:
  - 403010,
  - 403012,
  - 403014,
  - 403016, etc.

address (hex)	Memory data (hex)
40301C	FF
40301B	12
40301A	CA
403019	FE
403018	1B
403017	F0
403016	DE
403015	BC
403014	9A
403013	78
403012	56
403011	34
403010	12

Memory Alignment

- For 32-bit data, the address aligned at:
  - 403010,
  - 403014,
  - 403018,
  - 40301C, etc.

address (hex)	Memory data (hex)
40301C	FF
40301B	12
40301A	CA
403019	FE
403018	1B
403017	F0
403016	DE
403015	BC
403014	9A
403013	78
403012	56
403011	34
403010	12

Memory Alignment

- For 64-bit data, the address aligned at:
  - 403010,
  - 403018, etc.

address (hex)	Memory data (hex)
40301C	FF
40301B	12
40301A	CA
403019	FE
403018	1B
403017	F0
403016	DE
403015	BC
403014	9A
403013	78
403012	56
403011	34
403010	12

Software Architecture and x86

- **Intel x86** architecture views each memory location as one-byte (**byte-addressable**)
- **Intel x86** uses **little-endian** ordering system

address (hex)	Memory data (hex)
40301C	FF
40301B	12
40301A	CA
403019	FE
403018	1B
403017	F0
403016	DE
403015	BC
403014	9A
403013	78
403012	56
403011	34
403010	12

Variable Declaration

Pseudo-instruction	Description
DB	Declare initialize data **(8-bit)**
DW	Declare initialize data **(16-bit)**
DD	Declare initialize data **(32-bit)**
DQ	Declare initialize data **(64-bit)**
DT	Declare initialize data **(80-bit)** 80-bit extended floating point or 18-digit packed BCD 123_456_789_123_456_789p

Variable Declaration

These types of variables (db, dw, dd, dq, dt) are declared in:

section .data

Example:

```
section .data

var1 db 0x12
var2 dw 0x1234
var3 dd 0x12345678
```

Variable Declaration

Example:

```
section .data
var1 db 0x12, 0x34, 0x56, 0x78
section .text
MOV AL, [var1]
```

Label	Address	Data
	3	78
	2	56
	1	34
var1	0	12

Variable Declaration

Example:

```
section .data
var2 dw 0x1234, 0x5678
section .text
MOV AX, [var2]
```

Label	Address	Data
	3	56
	2	78
	1	12
var2	0	34

Variable Declaration

Example:

```
section .data
var3 dd 0x12345678
section .text
MOV EAX, [var3]
```

Label	Address	Data
	3	12
	2	34
	1	56
var3	0	78

Variable Declaration

Example:

```
section .data
var4 dq 0x12345678ABCDEF10
section .text
MOV RAX, [var4]
```

Label	Address	Data
	7	12
	6	34
	5	56
	4	78
	3	ab
	2	cd
	1	ef
var4	0	10

Variable Declaration

Pseudo-instruction	Description
RESB	Declare uninitialized data **(8-bit)**
RESW	Declare uninitialized data **(16-bit)**
RESD	Declare uninitialized data **(32-bit)**
RESQ	Declare uninitialized data **(64-bit)**
REST	Declare uninitialized data **(80-bit)** 80-bit extended floating point or 18-digit packed BCD 123_456_789_123_456_789p

Variable Declaration

These types of variables (db, dw, dd, dq, dt) are declared in:

section .bss

Example:

```
section .bss
var1 resb 4       ; reserve 4*8-bit of memory
var2 resw 2       ; reserve 2*16-bit of memory
var3 resd 1       ; reserve 1*32-bit of memory
```

Variable Declaration

Pseudo-instruction	Description
TIMES	Prefix that will cause the instruction to be assembled multiple times

Example:

```
section .data
var1 times 3 db 0
```

Equivalent to

```
section .data
var1 db 0
     db 0
     db 0
```

Equivalent to

```
section .data
var1 db 0,0,0
```

For further detail, please refer to **https://www.nasm.us/doc/nasmdoc3.html**