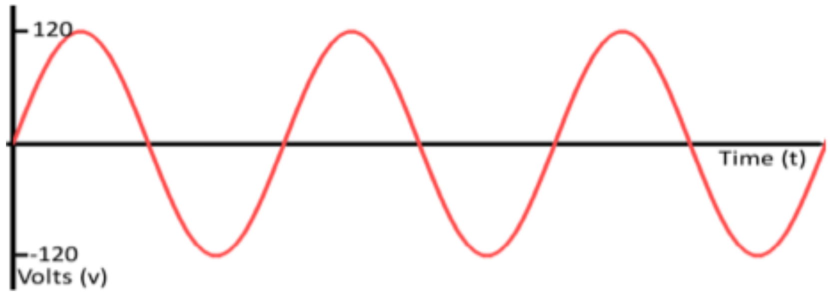
Bits, Bytes, Ints

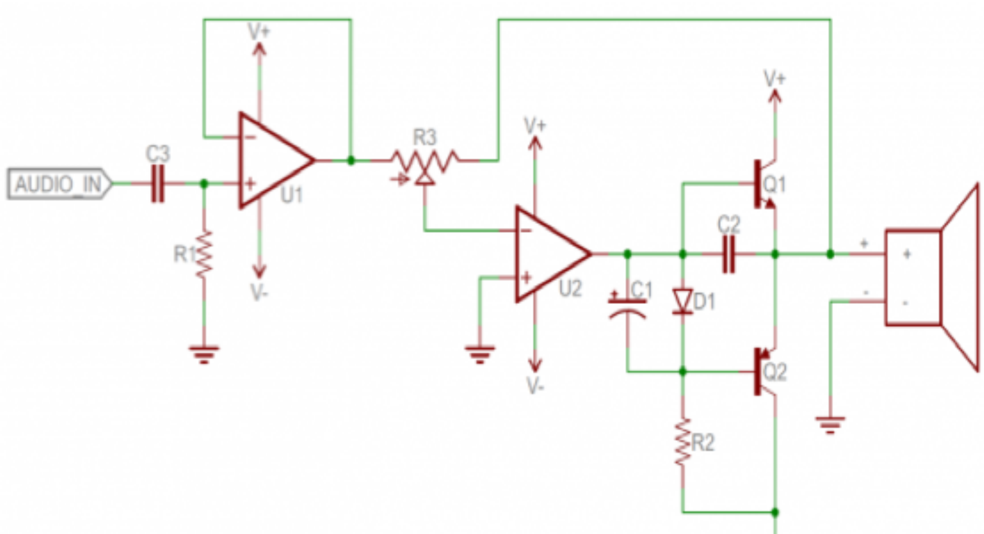
Shuai Mu

Slides are based on Tiger Wang's and Jinyang Li's class

The world has moved away from analog signal to …



Analog signals: smooth and continuous

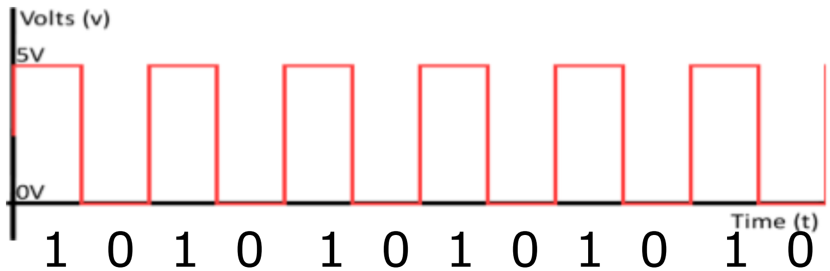


Problems

1. **Difficult to design**
2. **Susceptible to noise**

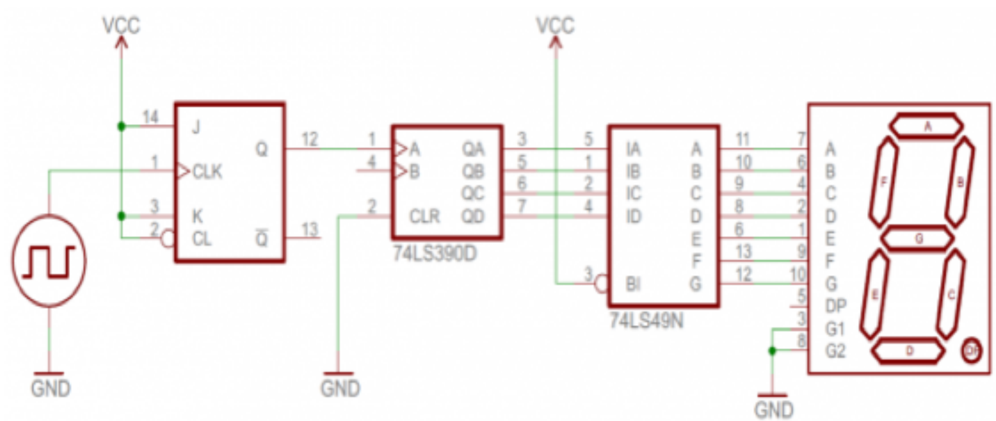
Analog components: resistors, capacitors, inductors, diodes, e.t.c…

… to digital



Digital signals: discrete (encode sequence of 0s and 1s)

Advantages



1. Easier to design
- Simple
- Integrate millions on a single chip

2. Reliable
- Robust to noise

Digital components: transistors, logic gates …

Using bits to represent everything

- Bit = Binary digit, 0 or 1
- A bit is too small to be used much
  - A bit has two values; the English alphabet has 26 value (characters)
- Using bits instead of bit
  - Group bits together
  - different possible bit patterns represent different "values"

Question

- How many values can a group of 2 bits represent?

0	**0**
0	1
1	0
1	1

4

- How many values can a group of n bits represent?

2^n

Allow us to represent
0, 1, 2, ... $(2^n - 1)$

Represent non-negative integer

bits: $b_{n-1}b_{n-2}\ldots b_2b_1b_0$

Question: how to map each bit pattern to a unique integer in $[0, 2^n - 1]$?

Solution: Base-2 representation

$$b_{n-1}b_{n-2}\ldots b_2b_1b_0 = \sum_{i=0}^{n-1} b_i * 2^i$$

b_i is bit at i-th position (from right to left, starting i=0)

Most significant bit (MSB)

The bit position has the greatest value

```
Bits    01010
MSB      ?

Bits    11011010
MSB      ?
```

Most significant bit (MSB)

The bit position has the greatest value

- The leftmost bit

```
Bits    01010
MSB      0

Bits    11011010
MSB      1
```

Least significant bit (LSB)

The bit position has the least value

– The rightmost bit

```
Bits    01010
MSB      0

Bits    11011010
MSB      0
```

Examples

Bits 0110

Value $0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = 6$

Bits 1110

Value ?

$$1*2^3+1*2^2+1*2^1+0*2^0 = 14$$

Byte

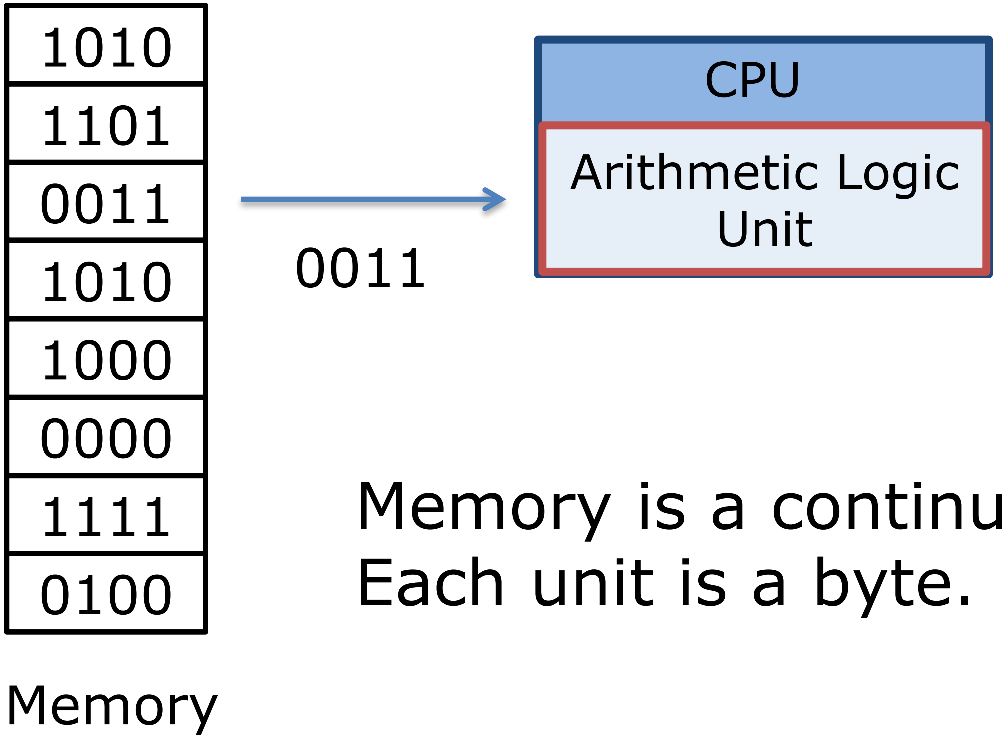
Each memory unit has multiple bits

- Dr. Werner Buchholz in July 1956
- Byte sizes from 1 bit to 48 bits have been used in the history

Byte

Each memory unit has multiple bits

- Dr. Werner Buchholz in July 1956
- Byte sizes from 1 bit to 48 bits have been used in the history

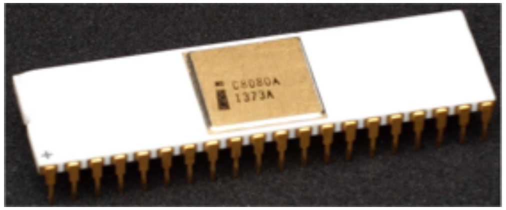


Memory is a continuous array.
Each unit is a byte.

Byte – 8 bits chunk

IBM System/360, 1964

Introduce



Intel 8080, 1974

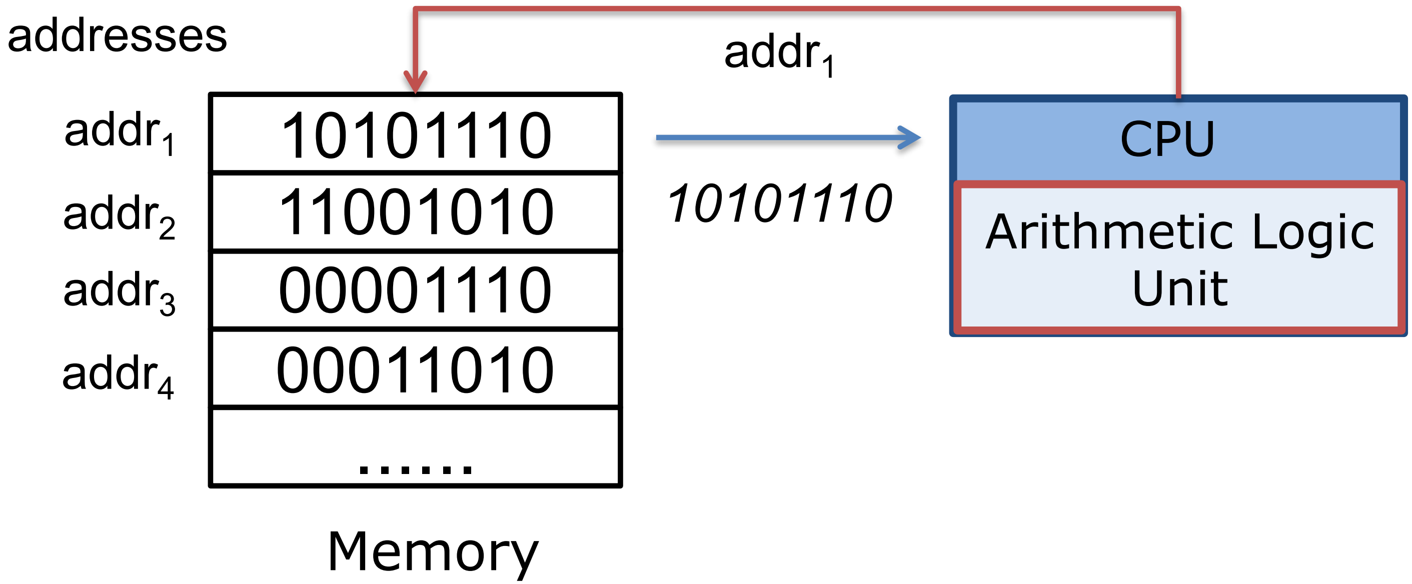
Widely adopted

Modern processors

Standard

Your mental model

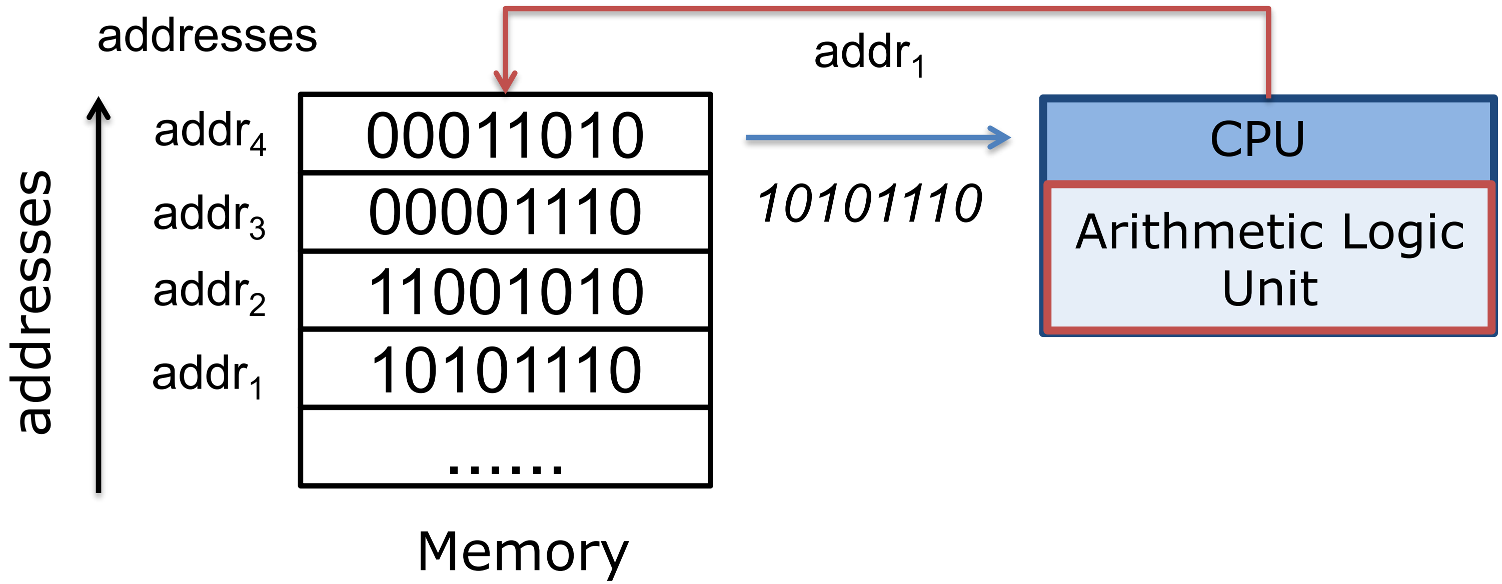
`Bits 10101110 11001010 00001110 00011010`



Each byte has 8 bits

Your mental model

Bits 10101110 11001010 00001110 00011010



Each byte has 8 bits

Range of Single Byte

Maximum

Minimum

Range of Single Byte

Maximum

- 11111111_2 → 255

Minimum

Range of Single Byte

Maximum

- 11111111_2 → 255

Minimum

- 00000000_2 → 0

Bit pattern description – intuitive way

Binary notation

```
Bits 10101110 11001010 00001110 00011010
```

4 bytes

Bit pattern description – intuitive way

Binary notation

- Too verbose

`Bits 10101110 11001010 00001110 00011010`

4 bytes

15 cm on my laptop

Bit pattern description – strawman

Decimal Notation

- how many decimal digits to represent one byte? 3

Bits	10101110	11001010	00001110	00011010
Decimal	174	202	014	026

Bit pattern description – strawman

Decimal Notation

– 3 decimal digits to represent one byte

```
Bits      10101110 11001010 00001110 00011010
Decimal   174      202      014      026
```

too tedious to do the conversion

Hexadecimal Notation

Write bit patterns as base-16 (hex) numbers

- Hex “digit” is one of 16 symbols: 0-9, a,b,c,d,e,f
- Each byte is two 4-bit chunks
- Each 4-bit-chunk is represented by a hex “digit”

Hexadecimal "digit"

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hexadecimal Notation

- Each byte is represented with 2 hex numbers (00_{16} -- FF_{16})

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex				
Hex(C)				

1010 =

Hexadecimal Notation

- Each byte is represented with 2 hex numbers (00_{16} -- FF_{16})

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A**			
Hex(C)				

$1010 = 1 * 2^3 + 0 * 2^2 + 1 * 2 + 0 = 10 = A_{16}$

Hexadecimal Notation

- Each byte is represented with 2 hex numbers (00_{16} -- FF_{16})

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A**			
Hex(C)				

1110 =

Hexadecimal Notation

- Each byte is represented with 2 hex numbers $(00_{16}$ -- $FF_{16})$

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A E**			
Hex(C)				

$1110 = 1 * 2^3 + 1 * 2^2 + 1 * 2 + 0 = 14 = E_{16}$

Hexadecimal Notation

- Each byte is represented with 2 hex numbers (00_{16} -- FF_{16})

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A E**			
Hex(C)				

1100 =

Hexadecimal Notation

- Each byte is represented with 2 hex numbers $(00_{16}$ -- $FF_{16})$

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A E**	**C**		
Hex(C)				

$1100 = 1 * 2^3 + 1 * 2^2 + 0 * 2 + 0 = 12 = C_{16}$

Hexadecimal Notation

- Each byte is represented with 2 hex numbers $(00_{16}$ -- $FF_{16})$

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A E**	**C**		
Hex(C)				

1010 =

Hexadecimal Notation

- Each byte is represented with 2 hex numbers $(00_{16}$ -- $FF_{16})$

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A E**	**C A**		
Hex(C)				

$1010 = 1 * 2^3 + 0 * 2^2 + 1 * 2 + 0 = 10 = A_{16}$

Hexadecimal Notation

- Each byte is represented with 2 hex numbers (00_{16} -- FF_{16})

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A E**	**C A**	**0**	
Hex(C)				

$0000 = 0 * 2^3 + 0 * 2^2 + 0 * 2 + 0 = 0 = 0_{16}$

Hexadecimal Notation

- Each byte is represented with 2 hex numbers (00_{16} -- FF_{16})

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A E**	**C A**	**0 E**	
Hex(C)				

$1110 = 1 * 2^3 + 1 * 2^2 + 1 * 2 + 0 = 14 = E_{16}$

Hexadecimal Notation

- Each byte is represented with 2 hex numbers (00_{16} -- FF_{16})

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A E**	**C A**	**0 E**	**1**
Hex(C)				

$0001 = 0 * 2^3 + 0 * 2^2 + 0 * 2 + 1 = 1 = 1_{16}$

Hexadecimal Notation

- Each byte is represented with 2 hex numbers (00_{16} -- FF_{16})

Bits	**10101110**	**11001010**	**00001110**	**00011010**
Decimal	**174**	**202**	**014**	**026**
Hex	**A E**	**C A**	**0 E**	**1 A**
Hex(C)				

$1010 = 1 * 2^3 + 0 * 2^2 + 1 * 2 + 0 = 10 = A_{16}$

Hexadecimal Notation

- Each byte is represented with 2 hex numbers $(00_{16} \text{ -- } FF_{16})$

Bits	10101110	11001010	00001110	00011010
Decimal	174	202	014	026
Hex	A E	C A	0 E	1 A
Hex(C)	0xAECA0E1A			

Exercises Time

Hexadecimal	Decimal	Binary
		1010 0111
		0011 1110
0xBC		
	55	
0xF3		

Answers

Hexadecimal	Decimal	Binary
0xA7	10*16 + 7 = 167	1010 0111
0x3E	3*16 + 14 = 62	0011 1110
0xBC	11*16 + 12 = 188	1011 1100
0x37	3*16+7=55	0011 0111
0xF3	15*16+3=243	1111 0011

Unsigned addition

11 + 10 0xB + 0xA $00001011_2 + 00001010_2$

Unsigned addition

11 + 10 | 0xB + 0xA | $00001011_2 + 00001010_2$

```
    0 0 0 0 1 0 1 1
+   0 0 0 0 1 0 1 0
___________________
```

Unsigned addition

11 + 10   0xB + 0xA $00001011_2 + 00001010_2$

```
    0 0 0 0 1 0 1 1
+   0 0 0 0 1 0 1 0
-------------------
                  1
```

Unsigned addition

11 + 10   0xB + 0xA $00001011_2 + 00001010_2$

```
    0 0 0 0 1 0 1 1
+   0 0 0 0 1 0 1 0
-------------------
                2 1
```

Unsigned addition

11 + 10 | 0xB + 0xA | $00001011_2 + 00001010_2$

$$
\begin{array}{cccccccccc}
 & & & & & & & 1 & \\
 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
+ & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
\hline
 & & & & & & & 0 & 1
\end{array}
$$

Unsigned addition

11 + 10

0xB + 0xA

$00001011_2 + 00001010_2$

```
          1
  0 0 0 0 1 0 1 1
+ 0 0 0 0 1 0 1 0
-----------------
              1 0 1
```

Unsigned addition

11 + 10

0xB + 0xA

$00001011_2 + 00001010_2$

$$
\begin{array}{r}
0\ 0\ 0\ 0\ 1\ 0\ 1\ 1 \\
+\quad 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0 \\
\hline
0\ 0\ 0\ 1\ 0\ 1\ 0\ 1
\end{array}
$$

Unsigned subtraction

11 - 10   0xB - 0xA $00001011_2 - 00001010_2$

$$
\begin{array}{r}
0\;0\;0\;0\;1\;0\;1\;1 \\
-\quad 0\;0\;0\;0\;1\;0\;1\;0 \\
\hline
0\;0\;0\;0\;0\;0\;0\;1
\end{array}
$$

Unsigned subtraction

11 - 10

0xB - 0xA

$00001011_2 - 00001010_2$

```
  0 0 0 0 1 0 1 1
- 0 0 0 0 1 0 1 0
-----------------
  0 0 0 0 0 0 0 1
```

Unsigned subtraction

10 - 11 | 0xA – 0xB | 00001010_2 - 00001011_2

$$
\begin{array}{r}
0\ 0\ 0\ 0\ 1\ 0\ 1\ 0 \\
-\quad 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1 \\
\hline
\end{array}
$$

? ? ?

Question:

How to represent negative numbers?

Strawman

Most significant bit (MSB) represent the sign

$0\ 0\ 0\ 0\ 0\ 0\ 0\ 1_2 \longrightarrow 1$

$1\ 0\ 0\ 0\ 0\ 0\ 0\ 1_2 \longrightarrow -1$

$$
\begin{array}{r}
0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\
+\quad 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\
\hline
\end{array}
$$

Strawman

Most significant bit (MSB) represent the sign

$0\ 0\ 0\ 0\ 0\ 0\ 0\ 1_2 \longrightarrow 1$

$1\ 0\ 0\ 0\ 0\ 0\ 0\ 1_2 \longrightarrow -1$

$$
\begin{array}{r}
0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\
+\quad 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\
\hline
1\ 0\ 0\ 0\ 0\ 0\ 1\ 0
\end{array}
$$

Strawman

Most significant bit (MSB) represent the sign

$0\,0\,0\,0\,0\,0\,0\,1_2 \longrightarrow 1$

$1\,0\,0\,0\,0\,0\,0\,1_2 \longrightarrow -1$

$$
\begin{array}{r}
0\,0\,0\,0\,0\,0\,0\,1 \\
+\quad 1\,0\,0\,0\,0\,0\,0\,1 \\
\hline
1\,0\,0\,0\,0\,0\,1\,0
\end{array}
$$

-2 ???

Two's complement

Byte 10010110

Unsigned number

$1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2 + 0 * 2^0$

Two's complement

Byte 10010110

Unsigned number

$1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2 + 0 * 2^0$

Signed number

$-1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2 + 0 * 2^0$

Two's complement

$$\vec{b} = \left[b_w, b_{w-1}, \ldots, b_0 \right]$$

$$val\left(\vec{b}\right) = -b_w 2^w + \sum_{i=0}^{w-1} b_i 2^i$$

MSB: $val(b_w) = -b_w * 2^w$

Other: $val (b_i) = b_i * 2^i$, $0 <= i < w$

Two's complement

$$\vec{b} = \left[b_w, b_{w-1}, \ldots, b_0 \right]$$

$$val\left(\vec{b}\right) = -b_w 2^w + \sum_{i=0}^{w-1} b_i 2^i$$

Binary	Value
1000 0001	
1010 0101	
0101 0101	

Two's complement

$$\vec{b} = \left[b_w, b_{w-1}, \ldots, b_0 \right]$$

$$val\left(\vec{b}\right) = -b_w 2^w + \sum_{i=0}^{w-1} b_i 2^i$$

Binary		Value
1000 0001	$-1 * 2^7 + 1$	-127
1010 0101		
0101 0101		

Two's complement

$$\vec{b} = \left[b_w, b_{w-1}, \ldots, b_0 \right] \qquad val\left(\vec{b}\right) = -b_w 2^w + \sum_{i=0}^{w-1} b_i 2^i$$

Binary		Value
1000 0001	$-1 * 2^7 + 1$	-127
1010 0101	$-1 * 2^7 + 2^5 + 2^2 + 2^0$	-91
0101 0101		

Two's complement

$$\vec{b} = \left[b_w, b_{w-1}, \ldots, b_0 \right] \qquad val\left(\vec{b}\right) = -b_w 2^w + \sum_{i=0}^{w-1} b_i 2^i$$

Binary		Value
1000 0001	$-1 * 2^7 + 1$	-127
1010 0101	$-1 * 2^7 + 2^5 + 2^2 + 2^0$	-91
0101 0101	$2^6 + 2^4 + 2^2 + 2$	85

Two's complement

$$\vec{b} = \left[b_w, b_{w-1}, \ldots, b_0 \right]$$

$$val\left(\vec{b}\right) = -b_w 2^w + \sum_{i=0}^{w-1} b_i 2^i$$

```
    0 0 0 0 0 0 0 1
+   1 0 0 0 0 0 0 1
-------------------
    1 0 0 0 0 0 1 0
```

Two's complement

$$\vec{b} = \left[b_w, b_{w-1}, \ldots, b_0\right]$$

$$val\left(\vec{b}\right) = -b_w 2^w + \sum_{i=0}^{w-1} b_i 2^i$$

	0 0 0 0 0 0 0 1	2^0	1
+	1 0 0 0 0 0 0 1	$-1 * 2^7 + 2^0$	-127
	1 0 0 0 0 0 1 0	$-1 * 2^7 + 2^1$	-126

Find 2's complement quickly

With a negative number, how to give its binary representation? e.g. -40

Find 2's complement quickly

With a negative number, how to give its binary representation? e.g. -40

Step 1. represent 40 in binary

0010 1000

Find 2's complement quickly

With a negative number, how to give its binary representation? e.g. -40

Step 2. flip all bits

0010 1000 ⟶ 1101 0111

Find 2's complement quickly

With a negative number, how to give its binary representation? e.g. -40

Step 3. add 1

0010 1000 → 1101 0111 → 1101 1000

Find 2's complement quickly

With a negative number, how to give its binary representation? e.g. -40

Step 3. add 1

0010 1000 → 1101 0111 → 1101 1000

$$
\begin{array}{r}
0010\ 1000 \\
+\ 1101\ 0111 \\
\hline
1111\ 1111
\end{array}
$$

Find 2's complement quickly

With a negative number, how to give its binary representation? e.g. -40

Step 3. add 1

0010 1000 → 1101 0111 → 1101 1000

```
  0010 1000
  1101 0111
+         1
-----------
1 0000 0000
```

Why does this trick work

- What is $1111...11_2$ in 2's complement?

- $$\vec{b} + (\sim \vec{b}) = 11...11_2 = -1$$

b with bits flipped

$$-\vec{b} = (\sim \vec{b}) + 1$$

Exercise Time II

Hexadecimal	Decimal	Binary
0xce		
		1001 1100
	127	
	-128	
	-90	

Answers

Hexadecimal	Decimal	Binary
0xce	-50	1100 1110
0x9c	-100	1001 1100
0x7f	127	0111 1111
0x80	-128	1000 0000
0xa6	-90	1010 0110

Ranges

	Range	Min	Max
1 byte unsigned	[0, $2^8 - 1$]	0	255
1 byte signed			

Ranges

	Range	Min	Max
1 byte unsigned	$[0, 2^8 - 1]$	0	255
1 byte signed	$[-2^7, 2^7 - 1]$	-128	127

Min: 1000 0000

Max: 0111 1111

Overflow

	1 0 0 0 0 0 0 1	-127
+	1 0 0 0 0 0 0 1	-127

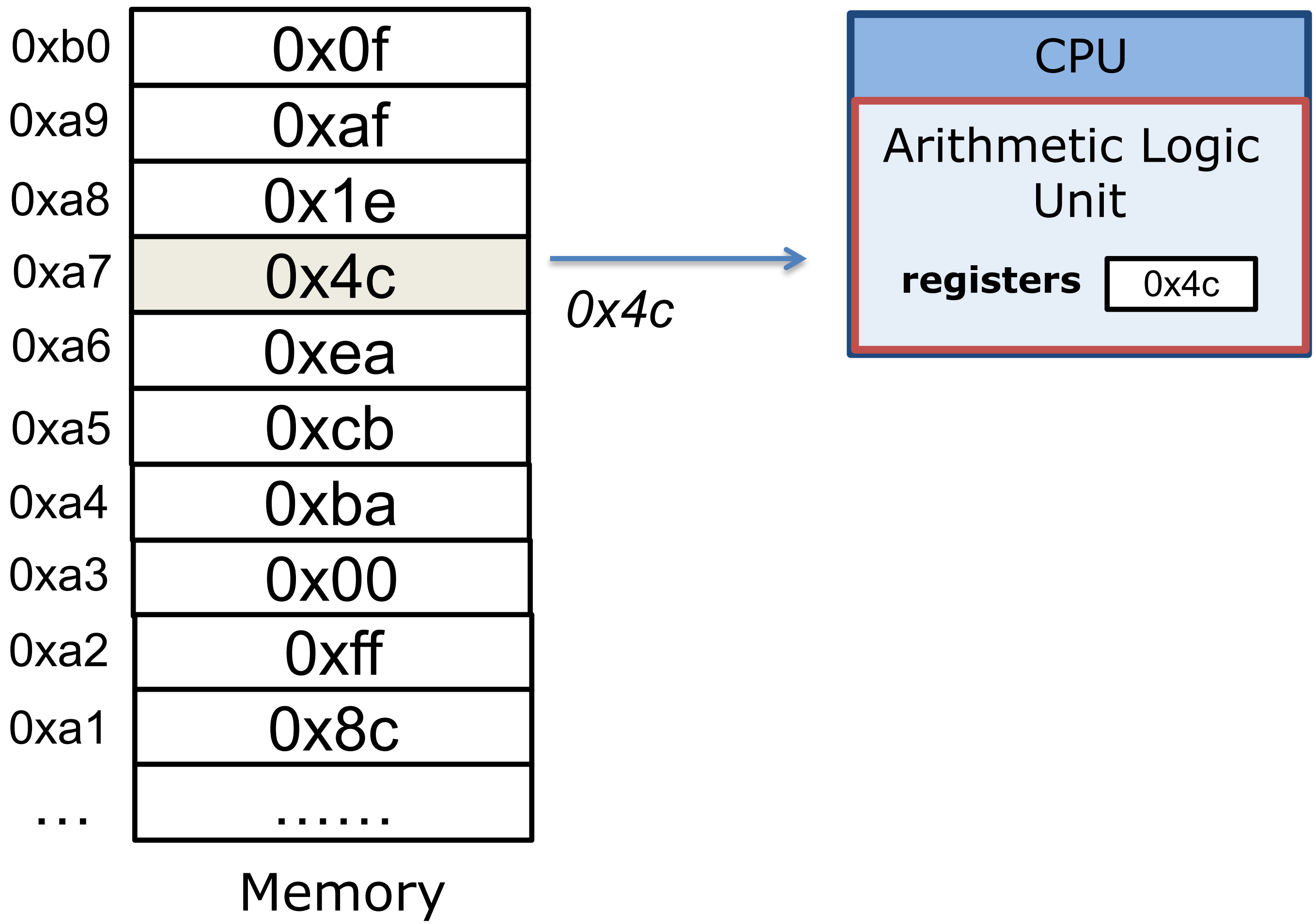
	Range	Min	Max
1 byte unsigned	$[0, 2^8 - 1]$	0	255
1 byte signed	$[-2^7, 2^7 - 1]$	-128	127

Overflow

	1 0 0 0 0 0 0 1	-127
+	1 0 0 0 0 0 0 1	-127
1	0 0 0 0 0 0 1 0	2 ???

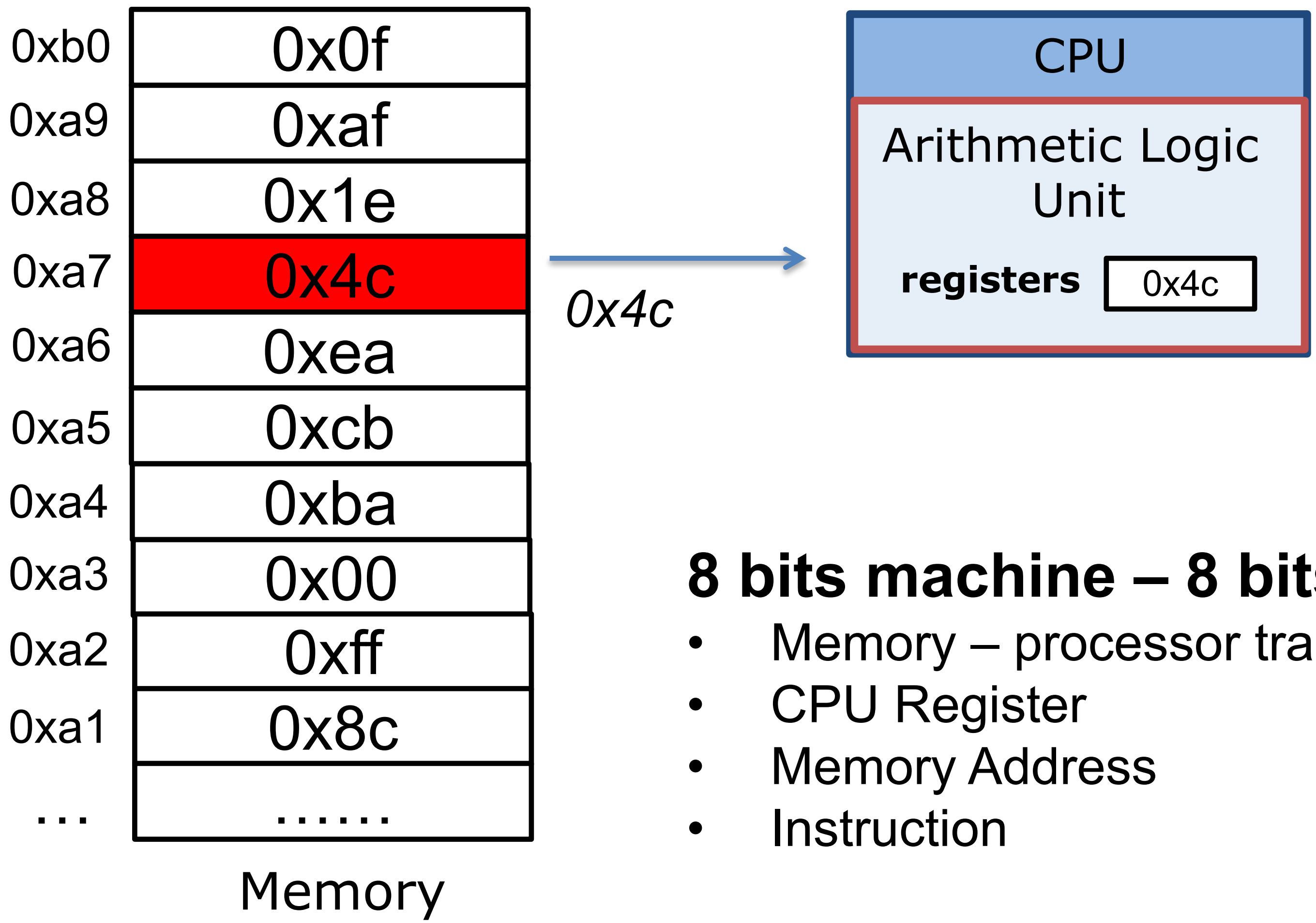
	Range	Min	Max
1 byte unsigned	$[0, 2^8 - 1]$	0	255
1 byte signed	$[-2^7, 2^7 - 1]$	-128	127

Intel 8080



0xb0
0x0f
0xa9
0xaf
0xa8
0x1e
0xa7
0x4c
0xa6
0xea
0xa5
0xcb
0xa4
0xba
0xa3
0x00
0xa2
0xff
0xa1
0x8c
...
......
Memory
0x4c
CPU
Arithmetic Logic Unit
registers
0x4c

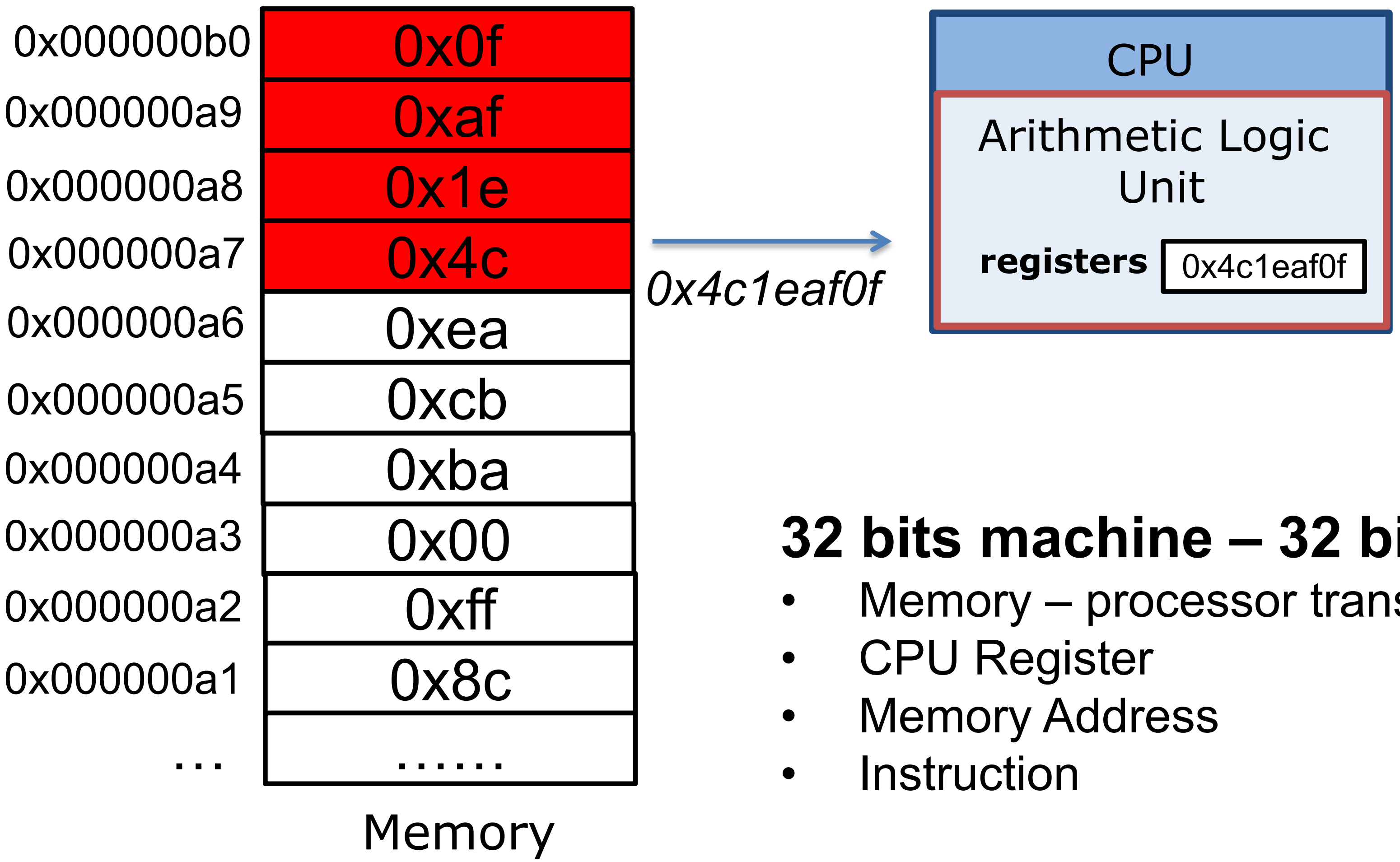
Intel 8080



8 bits machine – 8 bits length of

- Memory – processor transfer
- CPU Register
- Memory Address
- Instruction

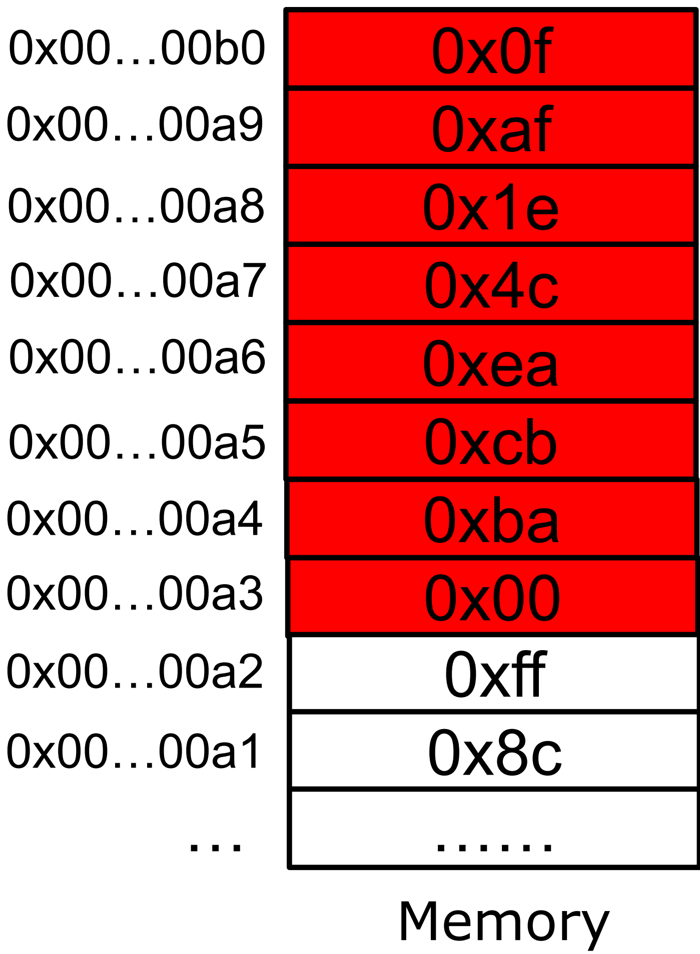
Intel 386

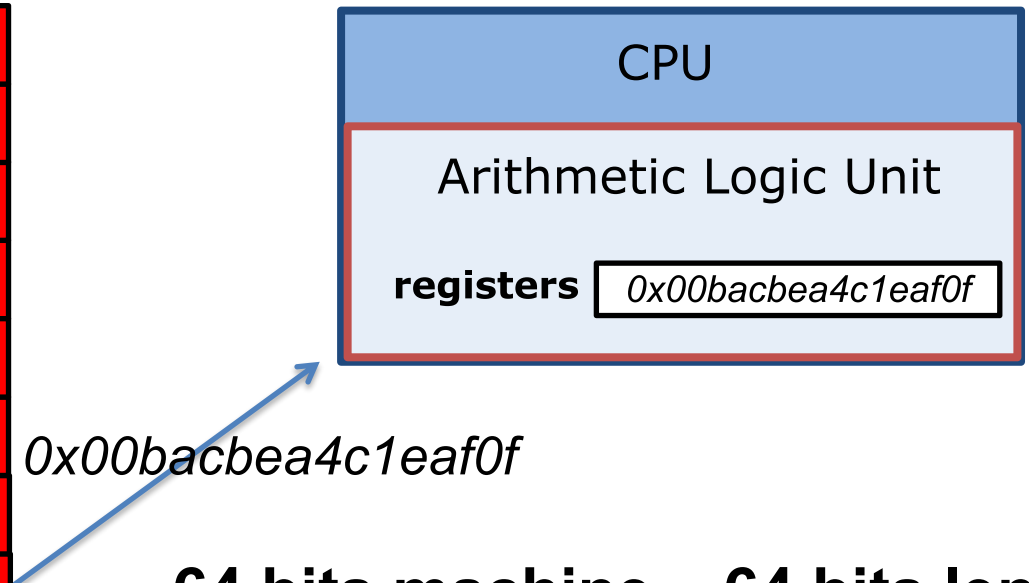


32 bits machine – 32 bits length of

- Memory – processor transfer
- CPU Register
- Memory Address
- Instruction

Intel Opteron → i7

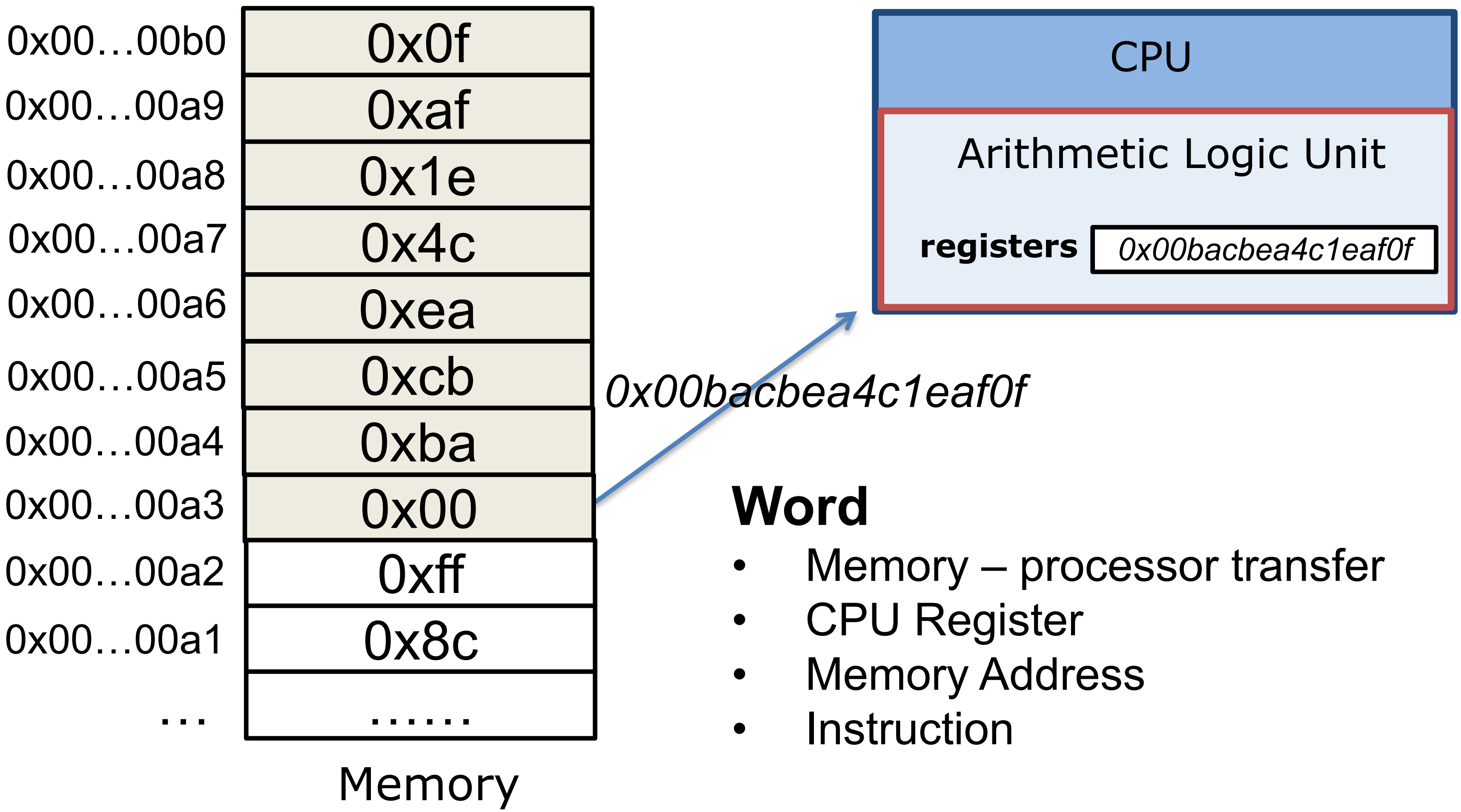




64 bits machine – 64 bits length of

- Memory – processor transfer
- CPU Register
- Memory Address
- Instruction

Intel Opteron → i7



Word

- Memory – processor transfer
- CPU Register
- Memory Address
- Instruction

Word

Definition

- Fixed size of data handled as a unit by the instruction set or processor

Length

- 8 for 8 bits machine
- 32 for 32 bits machine
- 64 for 64 bits machine

C's integral data types on 64 bits machine

	Length	Min	Max
[signed] char	1 byte	-2^7	$2^7 - 1$
unsigned char	1 byte	0	$2^8 - 1$
short	2 bytes		
unsigned short	2 bytes		
int	4 bytes		
unsigned int	4 bytes		
long	8 bytes		
unsigned long	8 bytes		

Integral data types on 64 bits machine

	Length	Min	Max
[signed] char	1 byte	-2^7	$2^7 - 1$
unsigned char	1 byte	0	$2^8 - 1$
short	2 bytes	-2^{15}	$2^{15} - 1$
unsigned short	2 bytes	0	$2^{16} - 1$
int	4 bytes		
unsigned int	4 bytes		
long	8 bytes		
unsigned long	8 bytes		

Integral data types on 64 bits machine

	Length	Min	Max
[signed] char	1 byte	-2^{7}	$2^{7} - 1$
unsigned char	1 byte	0	$2^{8} - 1$
short	2 bytes	-2^{15}	$2^{15} - 1$
unsigned short	2 bytes	0	$2^{16} - 1$
int	4 bytes	-2^{31}	$2^{31} - 1$
unsigned int	4 bytes	0	$2^{32} - 1$
long	8 bytes		
unsigned long	8 bytes		

Integral data types on 64 bits machine

	Length	Min	Max
[signed] char	1 byte	-2^{7}	$2^{7} - 1$
unsigned char	1 byte	0	$2^{8} - 1$
short	2 bytes	-2^{15}	$2^{15} - 1$
unsigned short	2 bytes	0	$2^{16} - 1$
int	4 bytes	-2^{31}	$2^{31} - 1$
unsigned int	4 bytes	0	$2^{32} - 1$
long	8 bytes	-2^{63}	$2^{63} - 1$
unsigned long	8 bytes	0	$2^{64} - 1$

Your first C program

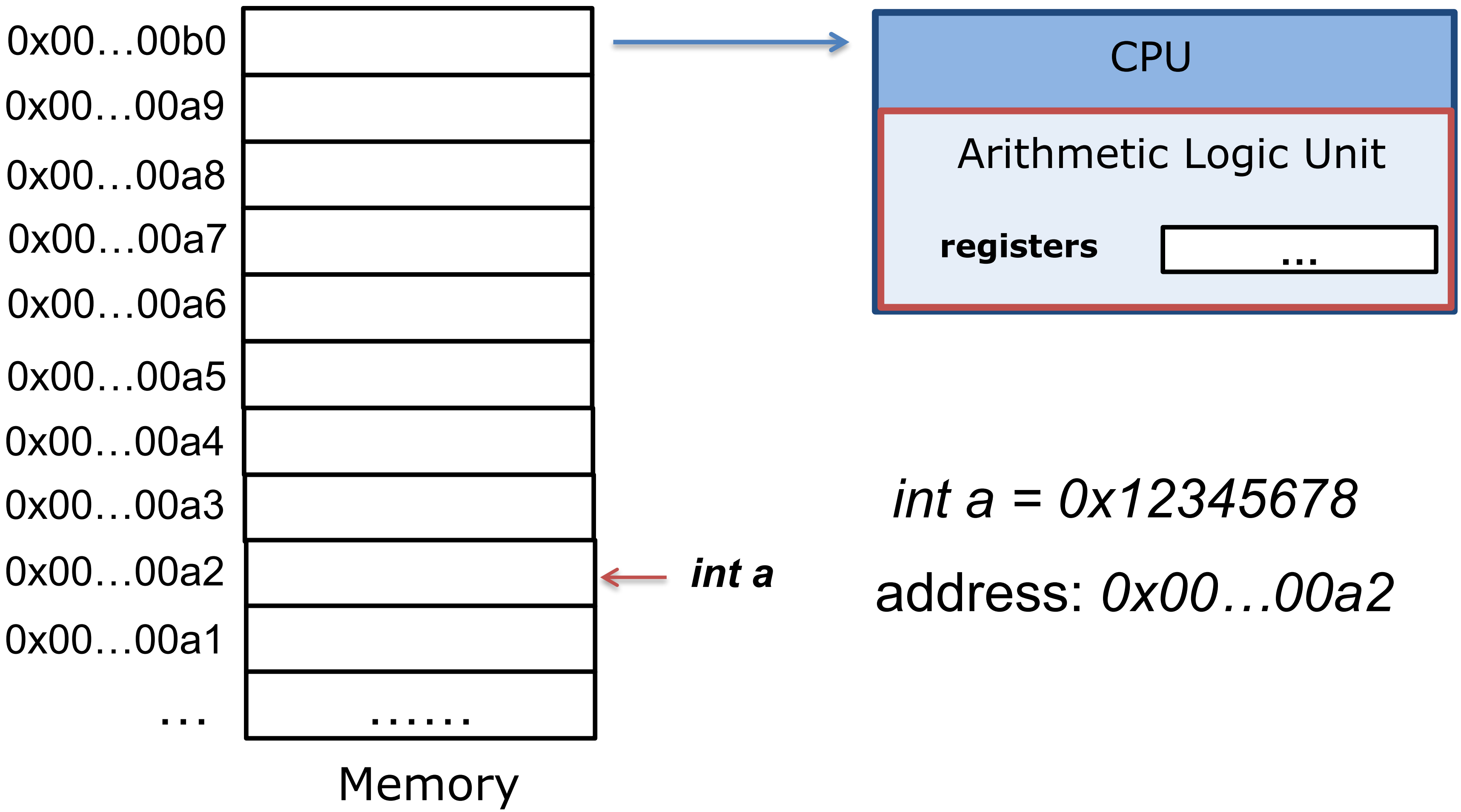
```
#include <stdio.h>

int
main()
{
      char x = -127;
      char y = 0x81;
      char z = x + y;
      printf("hello world sum is %d\n", z);
}
```

```
$ gcc helloworld.c
$ ./a.out
```

	1 0 0 0 0 0 0 1	-127
+	1 0 0 0 0 0 0 1	-127
1	0 0 0 0 0 0 1 0	2

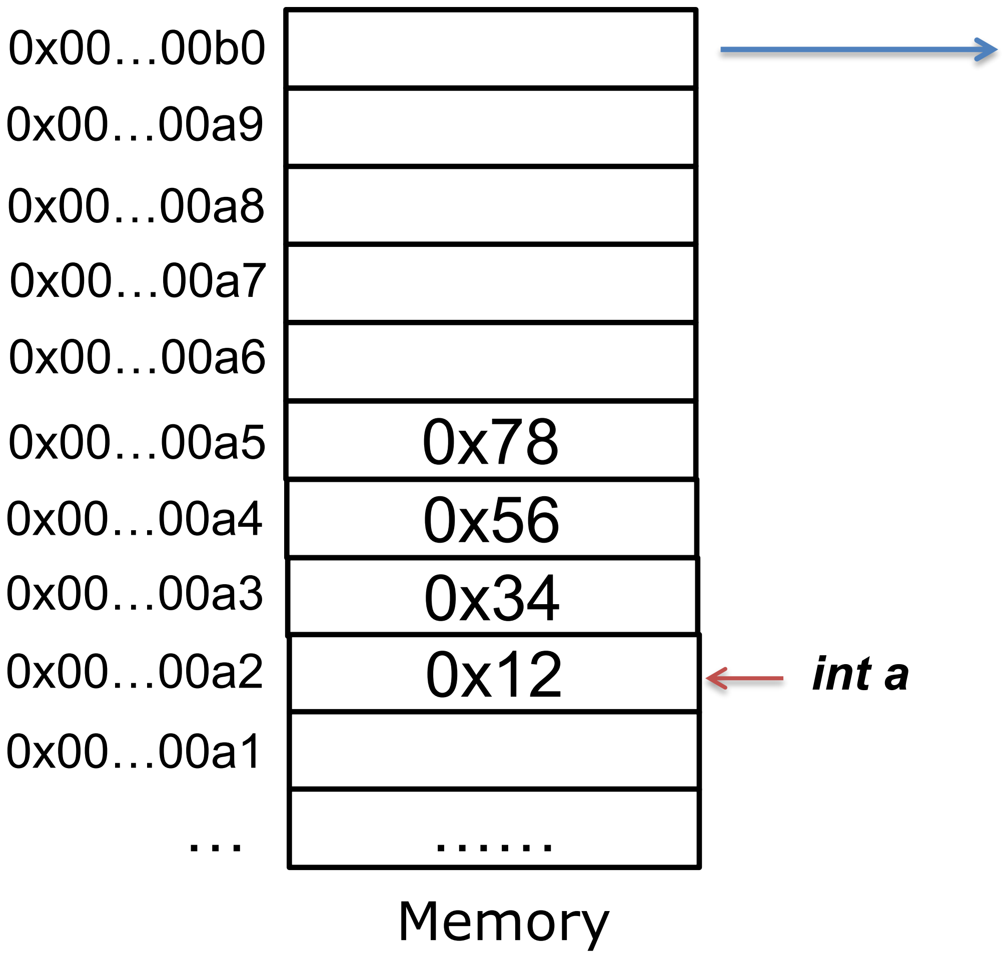
Memory layout

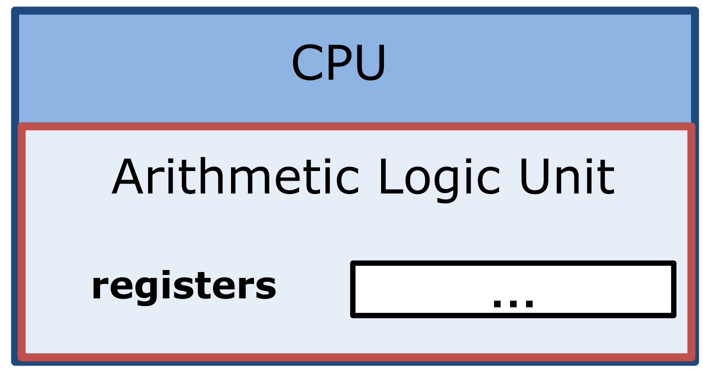


int a = 0x12345678

address: *0x00…00a2*

Memory layout – Intuition

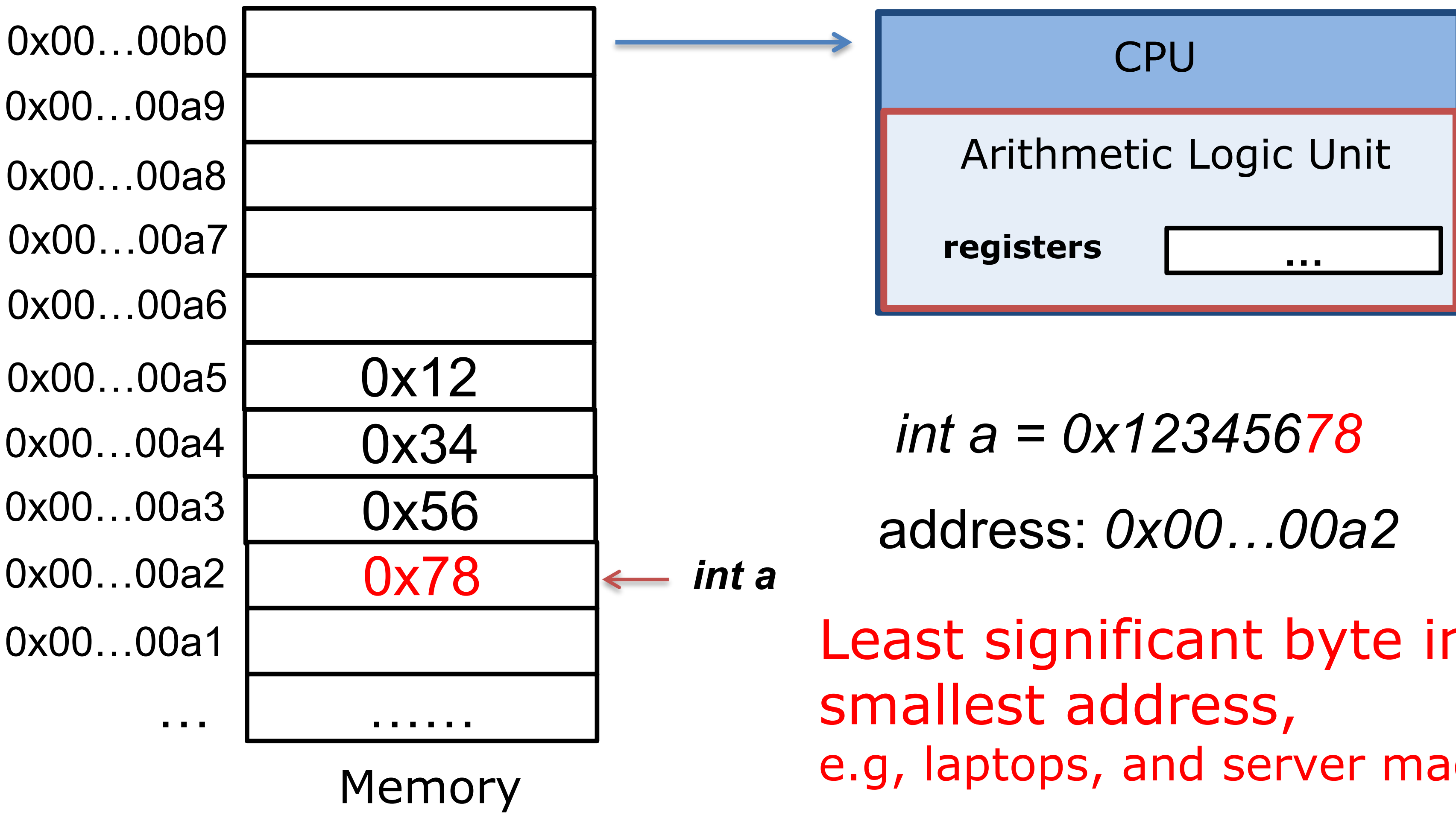




int a = 0x12345678

address: *0x00…00a2*

Memory layout – Little Endian



int a = 0x12345678

address: *0x00…00a2*

Least significant byte in smallest address,
e.g, laptops, and server machines

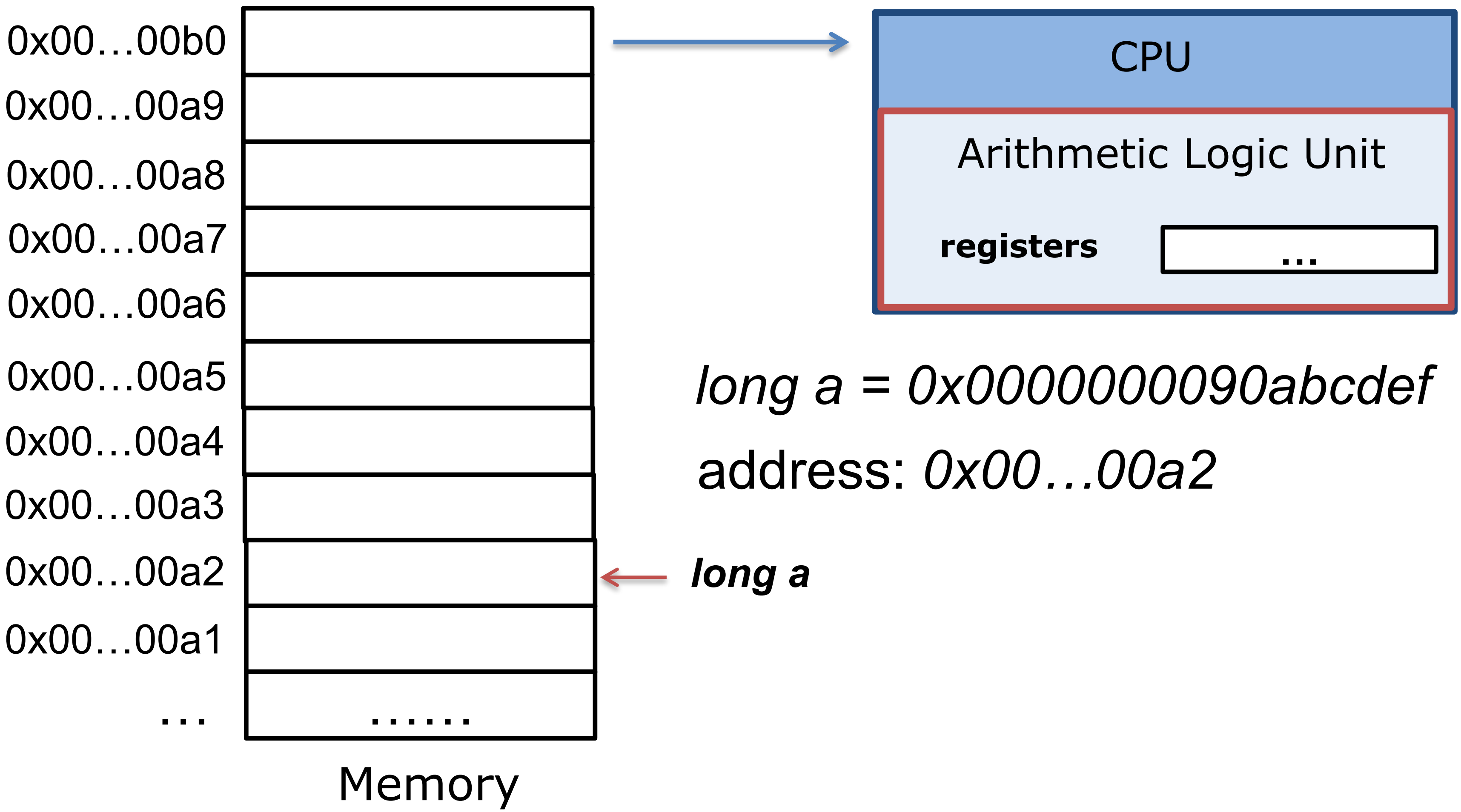
Advantages of Little Endian

```
  0x12345678
+ 0x12131415
```

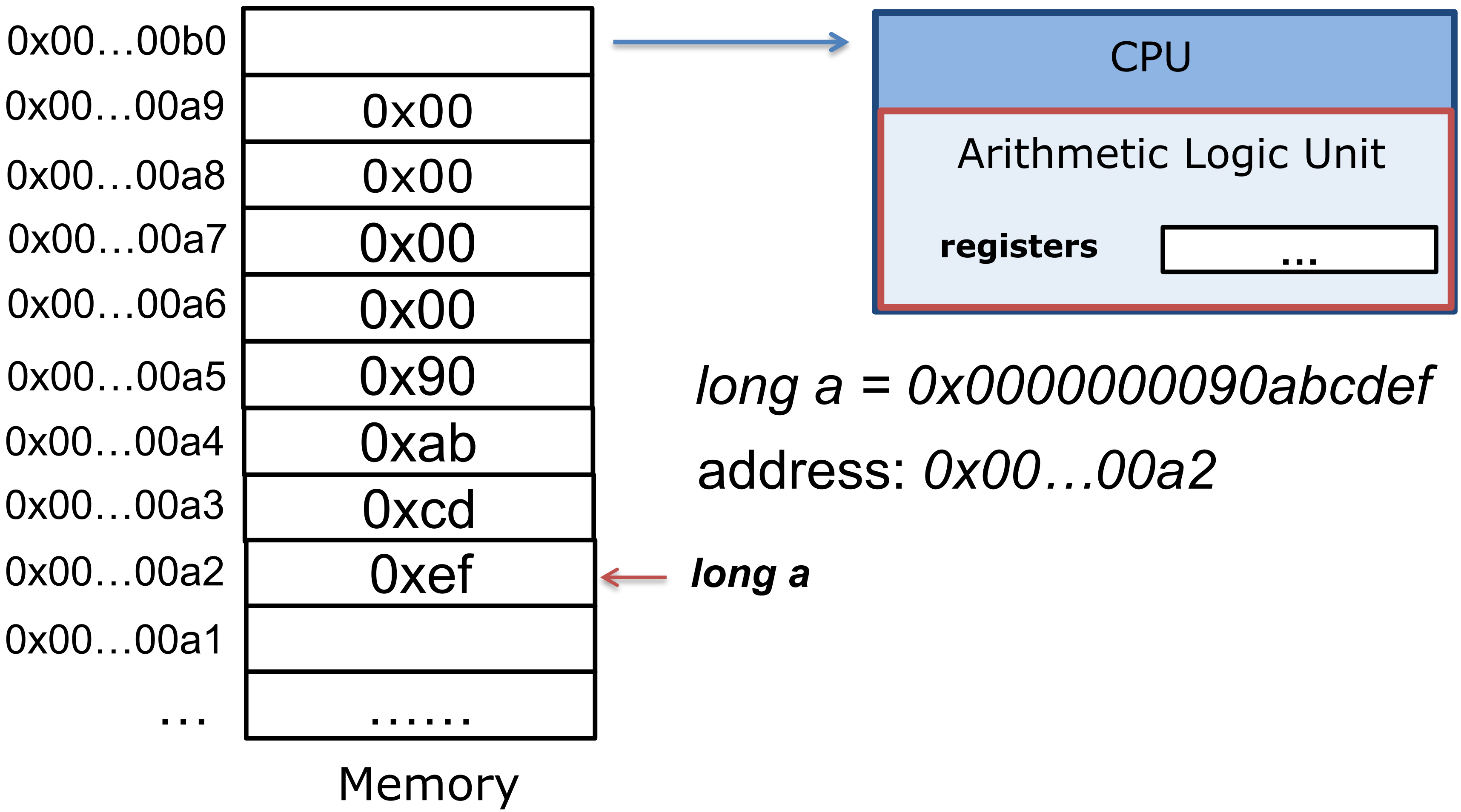
Processor performs the calculation from the least significant bit

Processor can simultaneously perform memory transfer and calculation.

Advantages of Little Endian



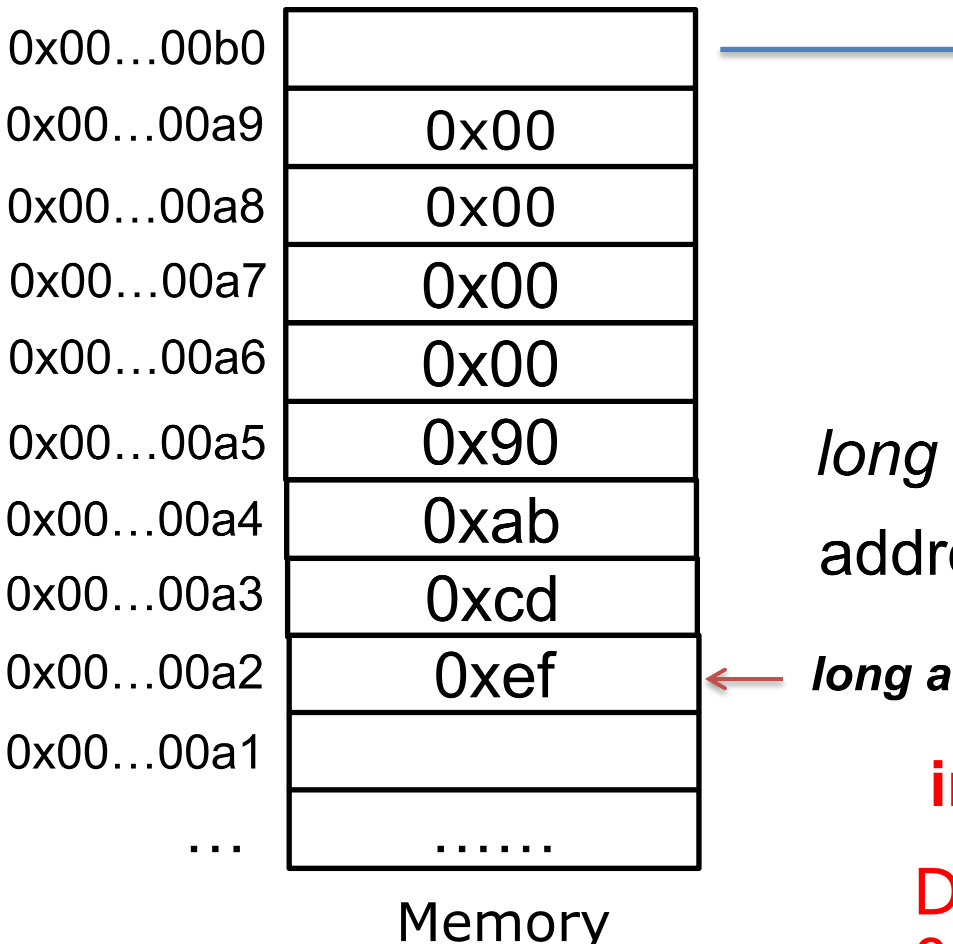
Advantages of Little Endian

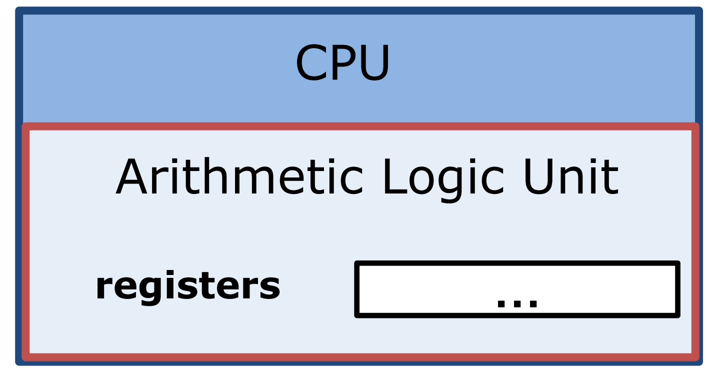


long a = 0x0000000090abcdef

address: *0x00…00a2*

Advantages of Little Endian





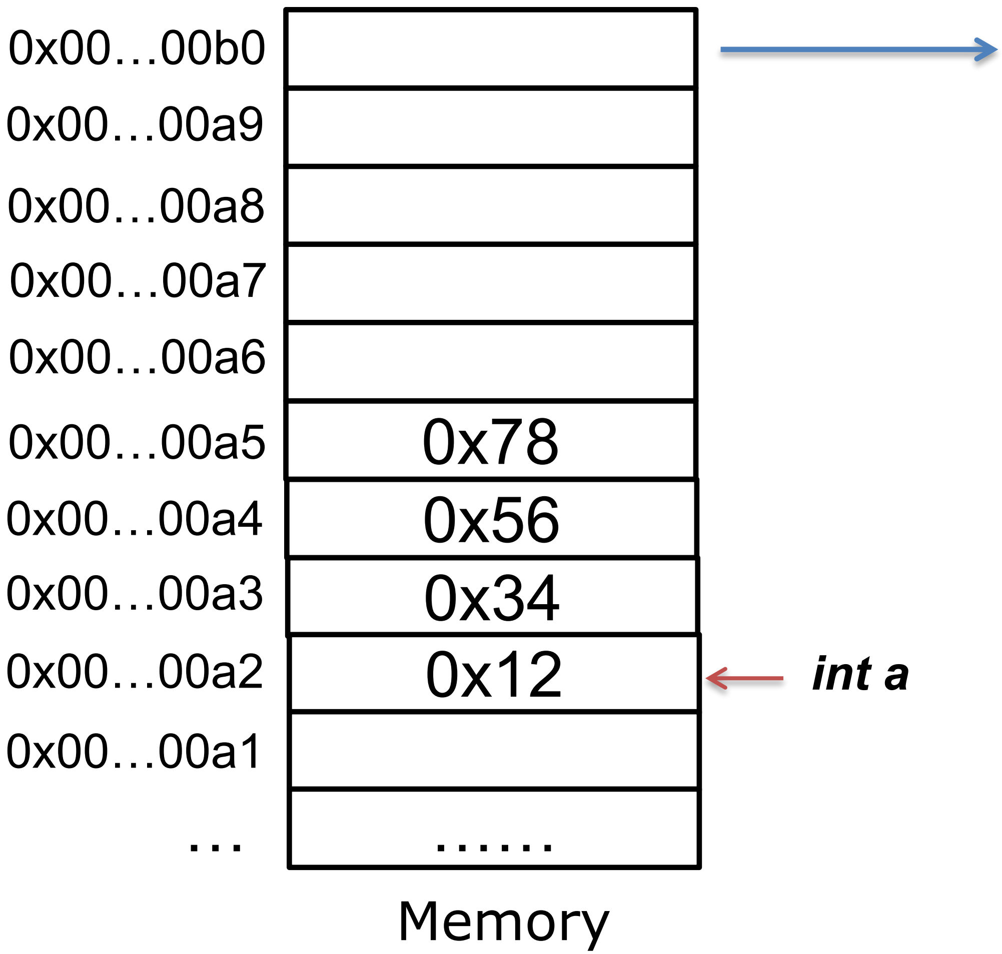
long a = 0x0000000090abcdef

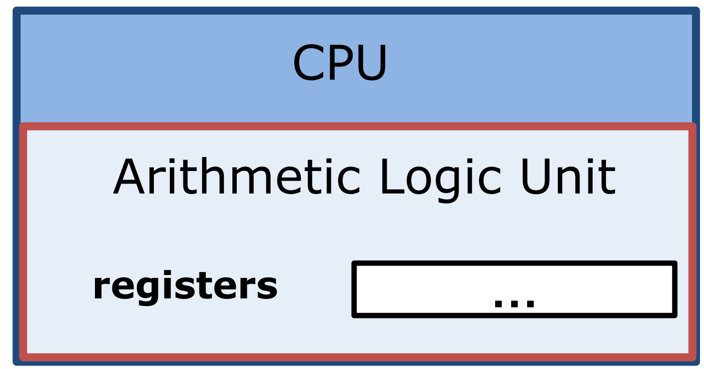
address: *0x00…00a2*

int b = (int)a

Directly get the data from 0x00…00a2

Memory layout – Intuition



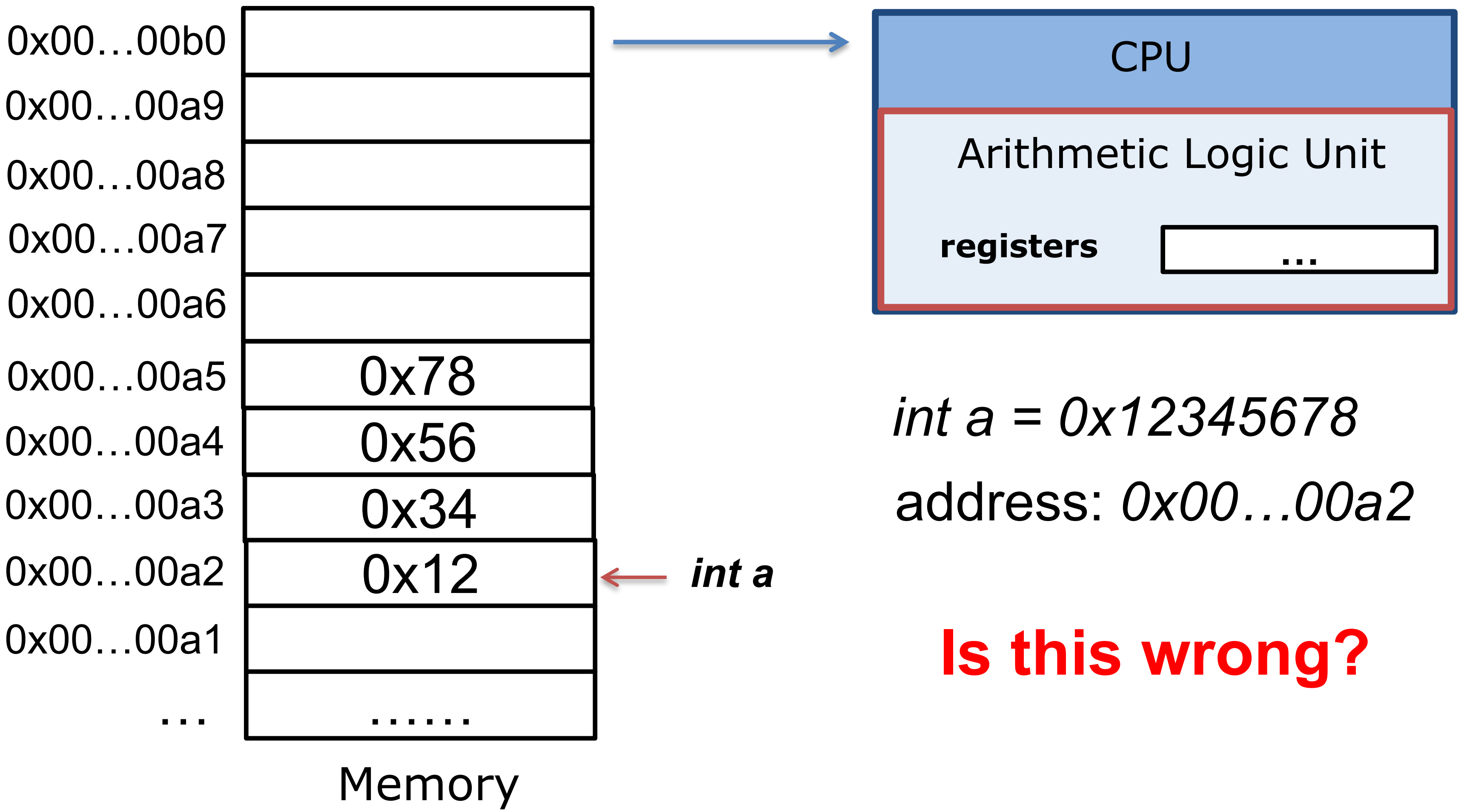


int a = 0x12345678

address: *0x00…00a2*

Is this wrong?

Memory layout – Intuition

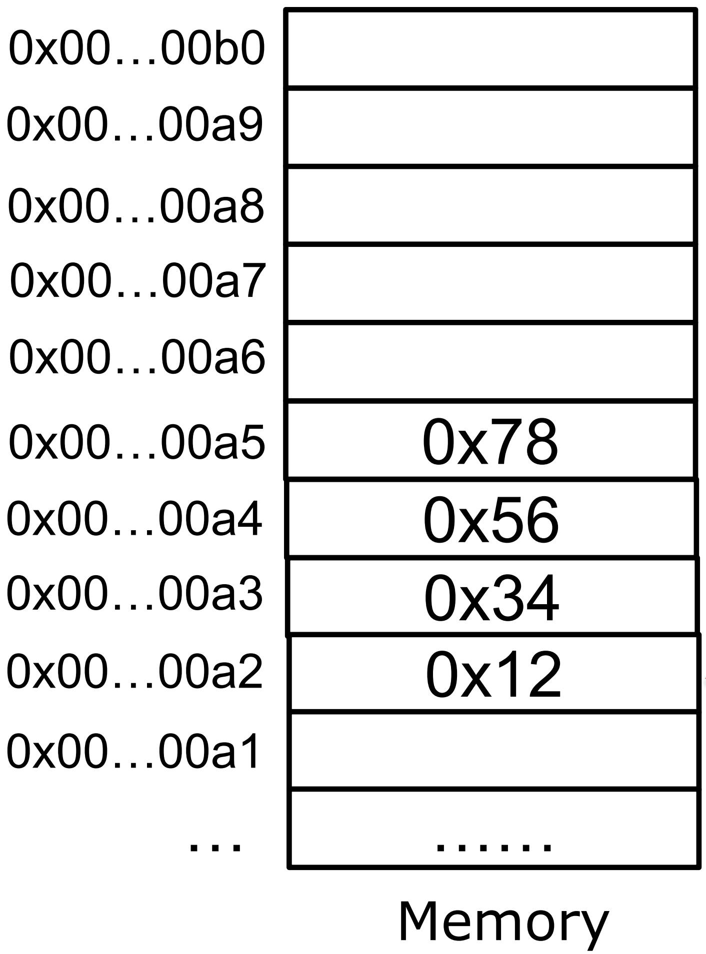


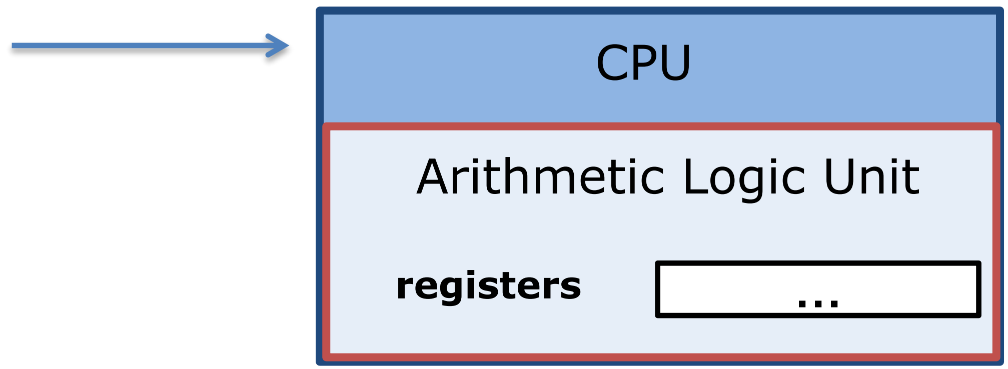
int a = 0x12345678

address: *0x00…00a2*

Is this wrong?

Memory layout – Big Endian





int a

int a = 0x12345678

address: *0x00…00a2*

Most significant byte in smallest address,

e.g, ARM architecture >v3 (cellphones, ipads)

Advantages of Big Endian

1. Easy to read

2. Test whether the number is positive or negative by looking at the byte at offset zero.