

A decorative graphic on the left side of the slide, consisting of a network of thin, light-blue lines and small circles, resembling a circuit board or a data network. The lines are vertical and horizontal, with some diagonal connections, and the circles are placed at various points along these lines.

2110104: COMPUTER PROGRAMMING

DATA REPRESENTATION

DEPT. OF COMPUTER ENGINEERING
CHULALONGKORN UNIVERSITY

Bit (Binary Digit)

- คอมพิวเตอร์เก็บข้อมูลแบบฐานสอง 0 กับ 1
- bit : หน่วยเล็กสุดของเลขฐานสอง 0 กับ 1
- byte : ลำดับของ bits จำนวน 8 bits
- ลำดับบิตอะไร จะแทนอะไร ขึ้นกับรูปแบบการเข้ารหัสที่ตกลงกัน
เช่น 01000001
 - ถ้ามองเป็นจำนวนเต็ม \rightarrow 65 (ฐานสิบ)
 - ถ้ามองเป็นตัวอักษร \rightarrow A

ระบบเลขฐานสิบ

9 8 1 3 . 2 4

$$9 \times 10^3 + 8 \times 10^2 + 1 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1} + 4 \times 10^{-2}$$

ระบบเลขฐานสอง

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

$$8 + 4 + 0 + 1 + 0.5 + 0.25$$

$$13.75_{10}$$

การเลื่อนบิต (shift)

00001	1
00010	2
00100	4
01000	8
10000	16

00101	5
01010	10
10100	20

10110	22
01011	11
00101	5

shift left 1 bit → คูณ 2

shift right 1 bit → หาร 2 ปัดเศษ

```
int m = 71;  
m = m >> 1;    // 35  
m = m << 2;    // 140
```

C++ source #2 ✎ ✕

A ▾  + ▾   

C++

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     int m = 71;
7     m = m << 1;
8 }
```

x86-64 gcc 13.2 (Editor #2) ✎ ✕

x86-64 gcc 13.2

Compiler options...

A ▾  ▾    + ▾ 

```
1 main:
2     push    rbp
3     mov     rbp, rsp
4     mov     DWORD PTR [rbp-4], 71
5     sal     DWORD PTR [rbp-4]
6     mov     eax, 0
7     pop     rbp
8     ret
```

Shift
Arithmetic
Left

C++

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     int m = 71;
7     m = m * 2;
8 }
```

A ▾  ▾    + ▾ 

```
1 main:
2     push    rbp
3     mov     rbp, rsp
4     mov     DWORD PTR [rbp-4], 71
5     sal     DWORD PTR [rbp-4]
6     mov     eax, 0
7     pop     rbp
8     ret
```

```
C++
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     int m = 71;
7     m = m << 2;
8 }
```

```
main:
1
2     push    rbp
3     mov     rbp, rsp
4     mov     DWORD PTR [rbp-4], 71
5     sal     DWORD PTR [rbp-4], 2
6     mov     eax, 0
7     pop     rbp
8     ret
```

```
C++
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     int m = 71;
7     m = m * 4;
8 }
```

```
main:
1
2     push    rbp
3     mov     rbp, rsp
4     mov     DWORD PTR [rbp-4], 71
5     sal     DWORD PTR [rbp-4], 2
6     mov     eax, 0
7     pop     rbp
8     ret
```

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     int m = 71;
7     m = m * 17;
8 }

```

```

1 main:
2     push    rbp
3     mov     rbp, rsp
4     mov     DWORD PTR [rbp-4], 71
5     mov     edx, DWORD PTR [rbp-4]
6     mov     eax, edx
7     sal     eax, 4
8     add     eax, edx
9     mov     DWORD PTR [rbp-4], eax

```

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     int m = 71;
7     m = m * 716;
8 }

```

```

1 main:
2     push    rbp
3     mov     rbp, rsp
4     mov     DWORD PTR [rbp-4], 71
5     mov     eax, DWORD PTR [rbp-4]
6     imul    eax, eax, 716
7     mov     DWORD PTR [rbp-4], eax
8     mov     eax, 0

```


การบวกเลขฐานสอง

$$\begin{array}{r} 111 \\ 00101001 \\ + 00011010 \\ \hline 01000011 \end{array}$$

จำนวนเต็มไม่ติดลบ (unsigned)

0	0	0	0	=	0	1	0	0	0	=	8
0	0	0	1	=	1	1	0	0	1	=	9
0	0	1	0	=	2	1	0	1	0	=	10
0	0	1	1	=	3	1	0	1	1	=	11
0	1	0	0	=	4	1	1	0	0	=	12
0	1	0	1	=	5	1	1	0	1	=	13
0	1	1	0	=	6	1	1	1	0	=	14
0	1	1	1	=	7	1	1	1	1	=	15

4 บิต เก็บค่า 0 ถึง 2^4-1

การแทนจำนวนเต็มลบ (2's complement)

เปลี่ยนบิต $0 \rightarrow 1, 1 \rightarrow 0$ แล้วบวกอีก 1

$$\begin{array}{r} 70_{10} \quad \boxed{0}100 \ 0110 \\ -70_{10} \quad \underline{1011 \ 1010} \\ \hline 0000 \ 0000 \end{array} + \begin{array}{r} 1011 \ 1001 \\ 1 \\ \hline \boxed{1}011 \ 1010 \end{array} +$$

-70_{10}

บิตซ้ายสุดเป็น 0 → บวก
เป็น 1 → ลบ

```
using namespace std;
```

```
int n1 = 0b000000000000000000000000000000001000110; // 70
```

[illegible]

```
int n3 = 0b1000110; // 70
```

```
int n4 = ~n3 + 1; // -70
```

```
cout << n1 << ' ' << n2 << ' ' << n3 << ' ' << n4;
```

2110104 COMPUTER PROGRAMMING

จำนวนเต็ม -1

เปลี่ยนบิต $0 \rightarrow 1$, $1 \rightarrow 0$ แล้วบวกอีก 1

$$\begin{array}{r} 0000 \ 0001 \\ 1111 \ 1110 \\ \quad \quad \quad + \\ \quad \quad \quad 1 \\ \hline 1111 \ 1111 \end{array}$$

จำนวนเต็มฐานสองขนาด 4 บิต

$$0\ 0\ 0\ 0 = 0$$

$$0\ 0\ 0\ 1 = 1$$

$$0\ 0\ 1\ 0 = 2$$

$$0\ 0\ 1\ 1 = 3$$

$$0\ 1\ 0\ 0 = 4$$

$$0\ 1\ 0\ 1 = 5$$

$$0\ 1\ 1\ 0 = 6$$

$$0\ 1\ 1\ 1 = 7$$

$$1\ 1\ 1\ 1 = -1$$

$$1\ 1\ 1\ 0 = -2$$

$$1\ 1\ 0\ 1 = -3$$

$$1\ 1\ 0\ 0 = -4$$

$$1\ 0\ 1\ 1 = -5$$

$$1\ 0\ 1\ 0 = -6$$

$$1\ 0\ 0\ 1 = -7$$

$$1\ 0\ 0\ 0 = -8$$

4 บิต เก็บค่า -2^3 ถึง 2^3-1

จำนวนเต็มฐานสองขนาด 8 บิต

$$0000 \ 0000 = 0$$

$$0000 \ 0001 = 1$$

$$0000 \ 0010 = 2$$

$$0000 \ 0011 = 3$$

$$0000 \ 0100 = 4$$

...

$$0111 \ 1100 = 124$$

$$0111 \ 1101 = 125$$

$$0111 \ 1110 = 126$$

$$0111 \ 1111 = 127$$

$$1111 \ 1111 = -1$$

$$1111 \ 1110 = -2$$

$$1111 \ 1101 = -3$$

$$1111 \ 1100 = -4$$

...

$$1000 \ 0100 = -124$$

$$1000 \ 0011 = -125$$

$$1000 \ 0010 = -126$$

$$1000 \ 0001 = -127$$

$$1000 \ 0000 = -128$$

8 บิต เก็บค่า -2^7 ถึง 2^7-1

เลือกประเภทข้อมูลให้เหมาะสม

Data Type	Size (in bytes)	
short	2 (16 bits)	-32768 to 32767
int	4 (32 bits)	-2^{31} to $2^{31} - 1$
long	4 (32 bits)	-2^{31} to $2^{31} - 1$
long long	8 (64 bits)	-2^{63} to $2^{63} - 1$
unsigned short	2 (16 bits)	0 to 65535
unsigned int	4 (32 bits)	0 to $2^{32} - 1$ (4,294,967,295)
unsigned long	4 (32 bits)	0 to $2^{32} - 1$
unsigned long long	8 (64 bits)	0 to $2^{64} - 1$
char	1 (8 bits)	-128 to 127
unsigned char	1 (8 bits)	0 to 255

Floating Point Numbers

123.5625

123.5625×10^0

12.35625×10^1

1.235625×10^2

0.1235625×10^3

...

$F \times 10^E$

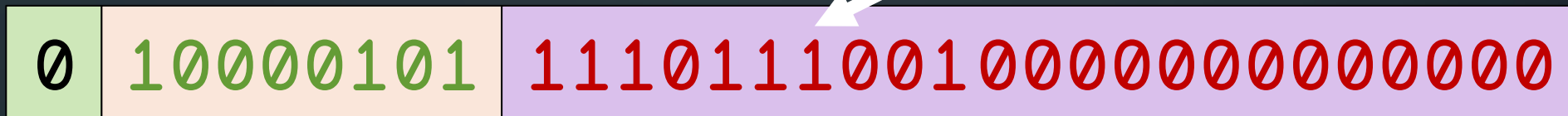
เก็บ F กับ E

แต่ 123.5625 แทนได้หลายแบบ
ใช้แบบไหนดี ?

32-bit single precision floating point

จำนวนลบ 1
ไม่ลบ 0

$$123.5625_{10} = 1111011.1001$$
$$= 1.1110111001 \times 2^6$$



sign of
mantissa

biased
exponent

normalized
mantissa

$$6 + 127 = 133_{10} = 10000101_2$$

IEEE-754

32-bit single precision floating point



$$10000001 = 129$$

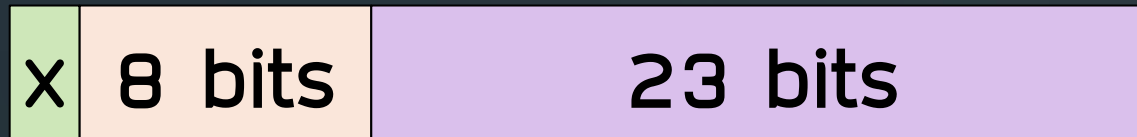
$$129 - 127 = 2 \quad 1.01010000000000000000000$$

$$-1.0101 \times 2^2 = 101.01_2 = 5.25_{10}$$

มีกรณีพิเศษ : 0, infinity, NAN, denormalized
(ขอไม่ลงรายละเอียด)

IEEE-754 floating point

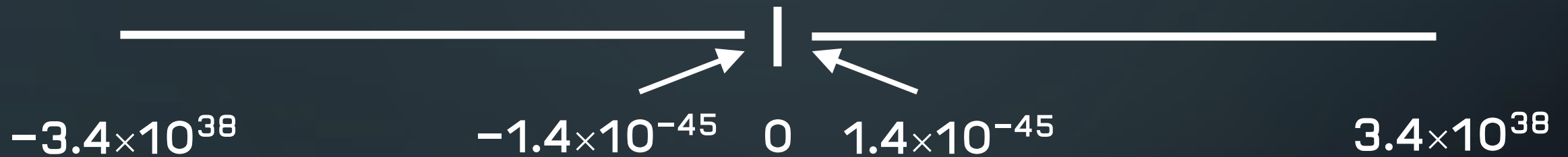
`float` : single-precision floating point



`double` : double-precision floating point



IEEE-754 single-precision floating point



ถ้าเทียบเป็นฐานสิบ ละเอียดยได้ประมาณ 7 หลัก

หมายเหตุ: ตัวเลขต่าง ๆ ข้างบนนี้เป็นค่าประมาณ

IEEE-754 double-precision floating point



ถ้าเทียบเป็นฐานสิบ ละเอียดได้ประมาณ 16 หลัก

หมายเหตุ: ตัวเลขต่าง ๆ ข้างบนนี้เป็นค่าประมาณ

เศษที่มีเลขฐานสองซ้ำไม่รู้จบ

$$0.1_{10} = 0.000110011 \dots = 0.0\overline{0011}$$

$$0.2_{10} = 0.00110011 \dots = 0.\overline{0011}$$

$$0.3_{10} = 0.0100110011 \dots = 0.01\overline{0011}$$

$$0.4_{10} = 0.01100110011 \dots = 0.011\overline{0011}$$

$$0.5_{10} = 0.1_2$$

```
double x = 0.1;  
double y = 3*x;  
cout << (y == 0.3); // 0
```

การเปรียบเทียบ floating point ว่า "เท่ากัน" มั้ย

~~$x = y$~~

~~$x \neq y$~~

$$\frac{|x - y|}{\max(|x|, |y|)} \leq \varepsilon \quad \varepsilon = 10^{-6}$$

$$|x - y| \leq \varepsilon(\max(|x|, |y|))$$

```
fabs(x-y) <= epsilon * max(fabs(x), fabs(y))
```


เลขฐานสิบหก

- มี 16 สัญลักษณ์
 - 0 1 2 3 4 5 6 7 8 9 A B C D E F
- ใช้เลขฐานสิบหก เพียงแค่ให้ อ่านเขียน เลขฐานสอง ได้กะทัดรัด
 - แบ่งเลขฐานสองเป็นกลุ่มละ 4 ตัว แล้วเขียนใหม่ด้วยฐานสิบหก

0100	1011	1110	1000	0001	0101	1111	0111
------	------	------	------	------	------	------	------

4	B	E	8	1	5	F	7
---	---	---	---	---	---	---	---

4B	E8	15	F7
----	----	----	----

การแทนอักขระ (characters)

character	binary	hex
A	0100 0001	41
B	0100 0010	42
C	0100 0011	43
D	0100 0100	44
E	0100 0101	45
F	0100 0110	46
G	0100 0111	47
H	0100 1000	48
I	0100 1001	49
J	0100 1010	4A

```
char c1 = 'A';  
char c2 = 0x4A;  
char c3 = c2 + 2;  
cout << c3; // L
```

ASCII codes

0000 0000 00

...

0111 1111 7F

hex	char	hex	char	hex	char	hex	char	hex	char	hex	char
20	SP	30	0	40	@	50	P	60	`	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	\$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	'	37	7	47	G	57	W	67	g	77	w
28	(38	8	48	H	58	X	68	h	78	x
29)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	
2D	-	3D	=	4D	M	5D]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	

```
lower = c | 0x20;  
upper = c & ~0x20;
```

```
if ('A' <= c && c <= 'Z')
```

```
if ('0' <= c && c <= '9')
```

```
if ('a' <= c && c <= 'z')
```

bit-wise operations

0	1	0	1	1	1	0	1
↓	↓	↓	↓	↓	↓	↓	↓
1	1	1	1	0	0	0	0
↓	↓	↓	↓	↓	↓	↓	↓
0	1	0	1	0	0	0	0

&

0	1	0	1	1	1	0	1
↓	↓	↓	↓	↓	↓	↓	↓
1	1	1	1	0	0	0	0
↓	↓	↓	↓	↓	↓	↓	↓
1	1	1	1	1	1	0	1

|

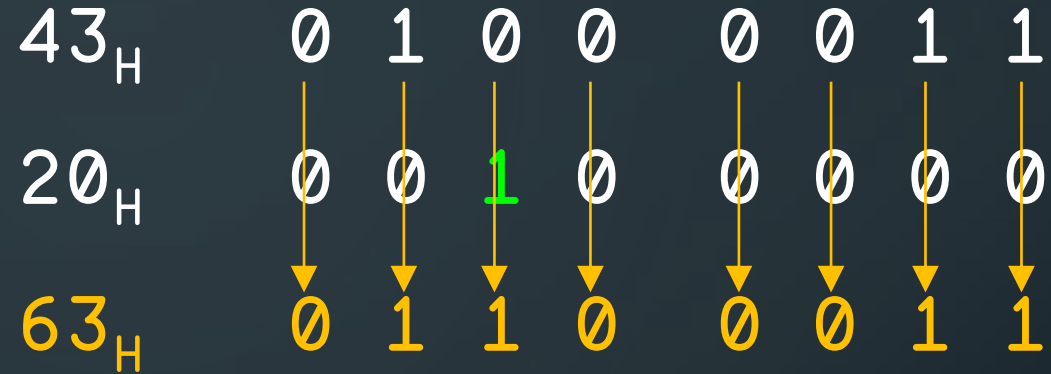
0	1	0	1	1	1	0	1
↓	↓	↓	↓	↓	↓	↓	↓
1	0	1	0	0	0	1	0

~

hex	char
40	@
41	A
42	B
43	C
44	D
45	E
46	F
47	G
48	H
49	I
4A	J
4B	K
4C	L
4D	M
4E	N
4F	O

hex	char
60	`
61	a
62	b
63	c
64	d
65	e
66	f
67	g
68	h
69	i
6A	j
6B	k
6C	l
6D	m
6E	n
6F	o

```
lower = ch | 0x20;
```



```
upper = ch & ~0x20;
```



char ก็เป็นจำนวนเต็ม

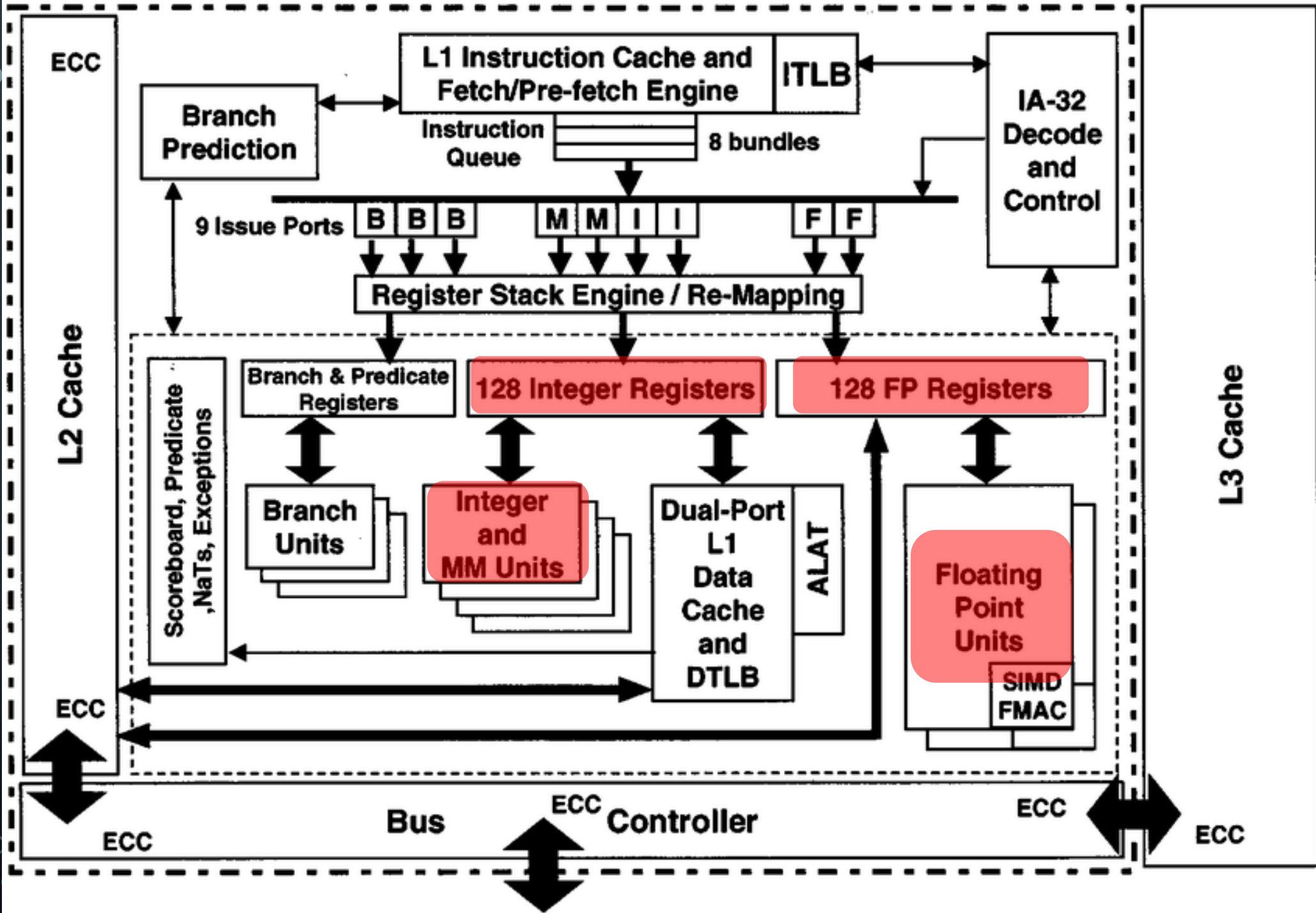
- $'A' + 1 = 0x41 + 1 = 65 + 1 = 66$
- $(char) ('A' + 1)$ ได้ 'B'
- $'A' + 'B' = 0x41 + 0x42 = 65 + 66 = 131$
- $'B' - 'A'$ ได้ 1, $'Z' - 'A'$ ได้ 25
- $'9' - '0'$ ได้ 9
- $(char)(9 + '0')$ ได้ '9'

UTF-8

- ASCII codes รหัส 00 – 7F ได้แค่ภาษาอังกฤษ
- Unicode รหัส 000000 – 10FFFFFF (1,114,112 codes)
- UTF-8 เป็นรหัส Unicode แบบจำนวนไบต์ไม่คงที่ขึ้นกับ code

Unicode (in hex)	UTF-8 (in binary)
00 – 7F	0xxxxxxx
080 – 7FF	110xxxxx 10xxxxxx
0800 – FFFF	1110xxxx 10xxxxxx 10xxxxxx
010000 – 1FFFFFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

- ก ไก่ U+0E01 → 0000 1110 0000 0001
- UTF-8 → 11100000 10111000 10000001 (E0 B8 81)



สตริง

- C-style string: เป็น array of chars ปิดท้ายด้วย `'\0'`
(สตริงมีอักขระ = N ตัว แต่ใช้อาร์เรย์ N+1 ช่อง)

```
char s1[] = {'H', 'e', 'l', 'l', 'o', '\0'};  
char s2[] = "Hello";
```

- C++ `std::string` เป็นการเก็บสตริงอีกแบบใน C++
(ข้อดี: มี methods ให้บริการ เช่น `size`, `substr`, `append`,...)

```
std::string s3 = "Hello";
```

- `"Hello"` ก็คือ `const char *` (ขอไม่ลงรายละเอียด)

สรุป

- เลือกประเภทข้อมูลให้เหมาะสมกับข้อมูลที่จะประมวลผล
- ข้อมูลที่เก็บใช้เนื้อที่จำกัด ตามประเภทข้อมูลที่ใช้ ทำให้
 - `int` ไม่ใช่จำนวนเต็ม (บวกไปเรื่อย ๆ ได้เลขลบ)
 - `float` ไม่ใช่จำนวนจริง ($0.1 * 3 \neq 0.3$ ได้ทั้ง)
- จัดสรรถูกเก็บด้วยการเข้ารหัส (ASCII, utf-8, ...)
- `char` คือจำนวนเต็ม ประมวลได้ด้วย `+` `-` `|` `&` `~` ...
- อยากรู้สตริง ใช้ `std::string` ง่ายดี