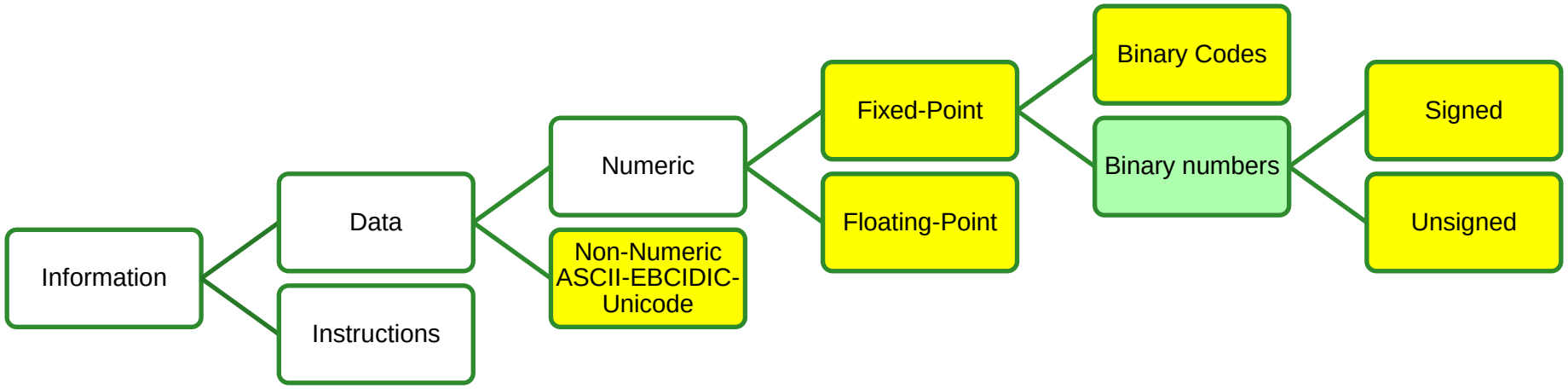
COMPUTER ARITHMETIC

Computer Arithmetic-I

Information - data

Binary numbers

Information - data



Numeric data

- Binary numbers
- Binary Codes
- Unsigned and Signed Magnitude numbers
- Fixed-Point numbers
- Floating-Point numbers.

Binary numbers

$(54.5)_{10} = (?)_2$

Binary numbers

$(54.5)_{10} = (110110.1)_2$

Why?

Binary numbers (**PDF notes-A**)

$(54.5)_{10} = (110110.1)_2$

2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
1	**1**	**0**	**1**	**1**	**0**	●	**1**	**0**	...

$(110110.1)_2 = 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1}$

$= 32 + 16 + 0 + 4 + 2 + 0 + 0.5$

$= 54.5$

Decimal-Binary-Octal-HEX

Dec	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	
11	
12	
13	
14	
15	

Decimal-Binary-Octal-HEX

Dec	Binary	Octal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	
9	1001	
10		
11		
12		
13		
14		
15		

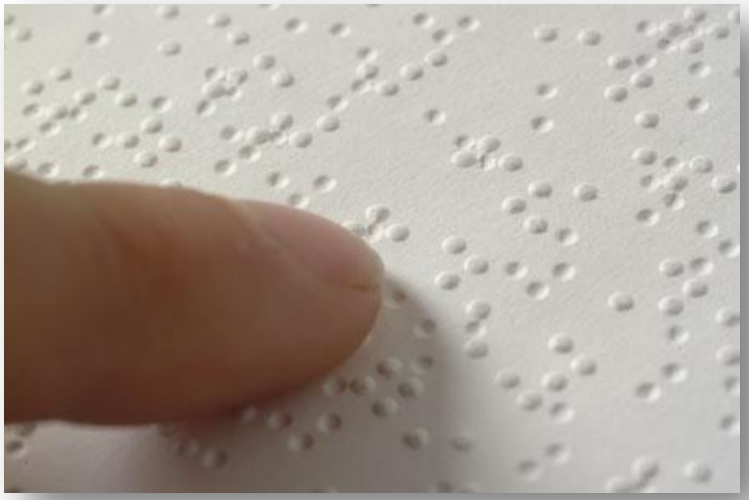
Decimal-Binary-Octal-HEX

Dec	Binary	Octal	Hex
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

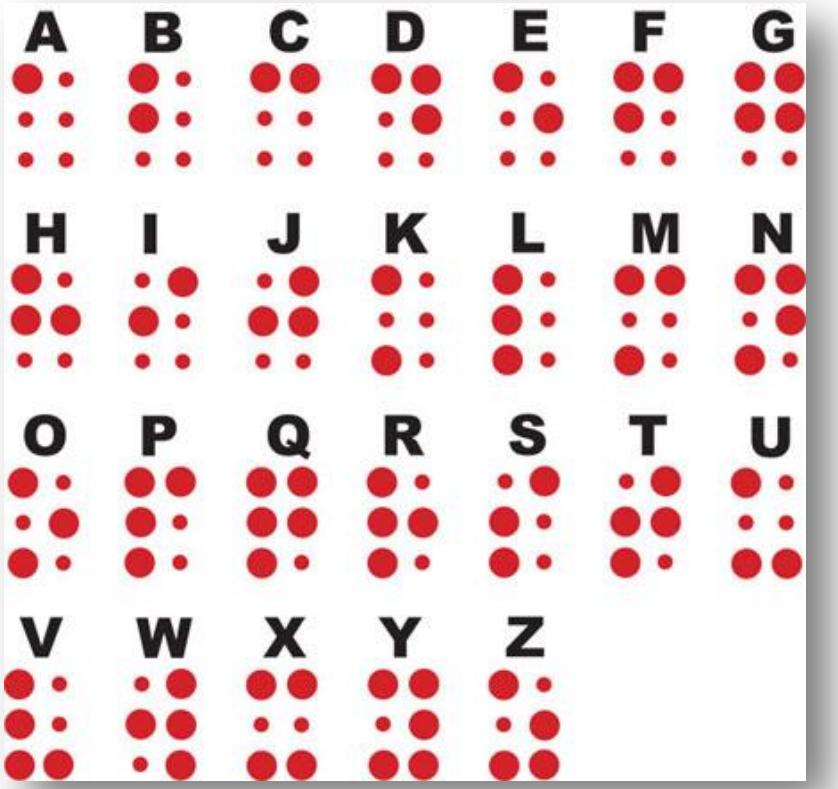
Codes …

Code used by people, with low vision, to read

Braille



Combinations of raised dots



Special Codes

- 8-4-2-1
- BCD
- Excess-3
- Aiken
- 2-out-of-5
- …

Binary (8-4-2-1)

	8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Binary Coded Decimal (BCD)

	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Excess-3 code (George Stibitz code)

	XS3
	0000
	0001
	0010
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100
	1101
	1110
	1111

Excess-3

	XS3
	0000
	0001
	0010
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100
	1101
	1110
	1111

Symmetric code

Aiken code

	Aiken
0	0000
1	0001
2	0010
3	0011
4	0100
	0101
	0110
	0111
	1000
	1001
	1010
5	1011
6	1100
7	1101
8	1110
9	1111

2-4-2-1 Code …

Aiken code

	Aiken
0	0000
1	0001
2	0010
3	0011
4	0100
	0101
	0110
	0111
	1000
	1001
	1010
5	1011
6	1100
7	1101
8	1110
9	1111

Symmetric code

2-4-2-1 Code …

2-out-of-5

	2-out-of-5
0	11000
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

It is used as Error Detecting Code; … natural even parity

2-out-of-5

	2-out-of-5
0	11000
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

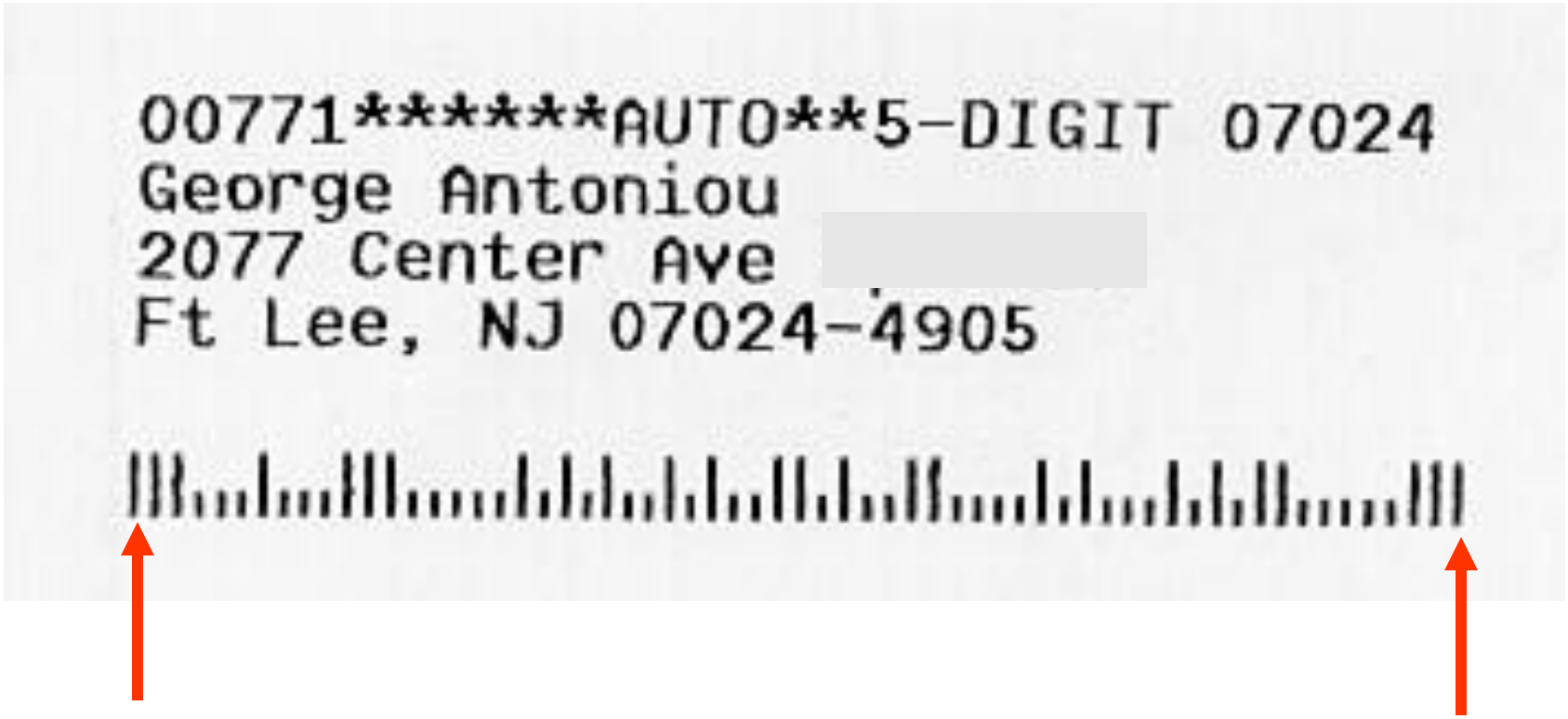
Except for the zero (0) which is decimal 24. The rest of the code follows the weights: 74210

2-out-of-5 (POSTNET)

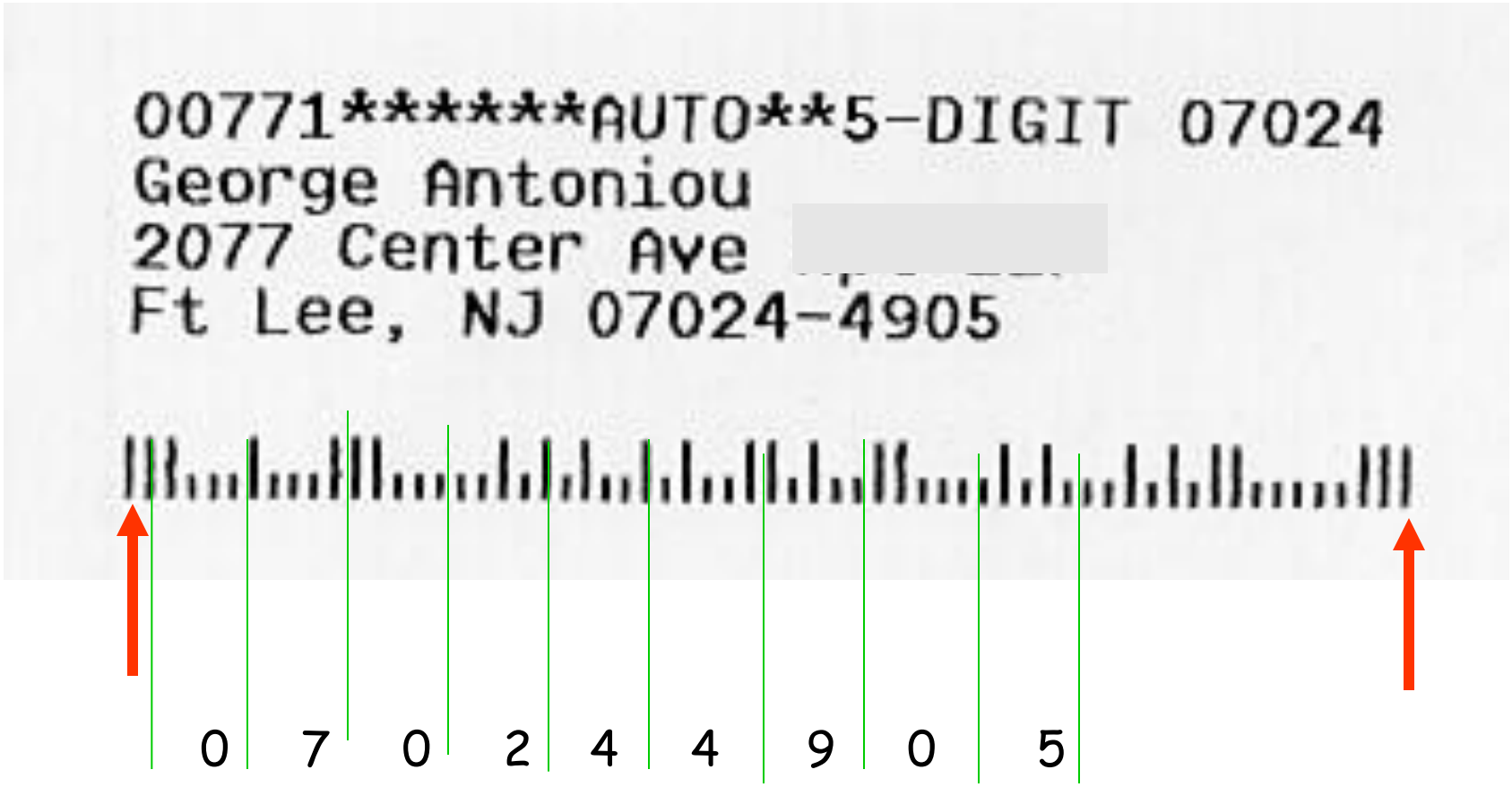
POSTNET (**P**ostal **N**umeric **E**ncoding **T**echnique) is a barcode symbology used by the United States Postal Service to assist in directing mail. (Wikipedia)

Today: 4-State Customer Barcode (4CB)

The first and last bars are the frame bars, used for aligning the scanner which reads the bar-code. The last digits are used for error correction (checksum)…..

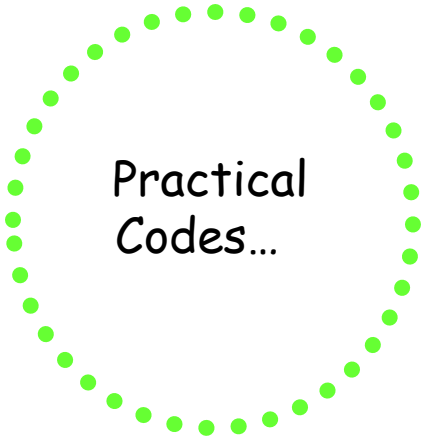


The first and last bars are the frame bars, used for aligning the scanner which reads the bar-code. The last digits are used for error correction (checksum)…..



Bar Code based codes

- **1-D codes**
- **2-D codes**



1-D and 2-D Codes **will not** be included in the exams

1-D codes

- **Code 39 = Code 3-of-9**; alphanumeric (full ASCII) bar code;
  applications in inventory, asset tracking, ID badges
- **Interleaved 2 of 5** ; Numeric-only bar code,
  industrial applications, carton labeling, laboratory uses
- **UPC (Universal Product Code),** Numeric Bar code;
  used in retail product labeling.
- **Code 128** – Alphanumeric (full ASCII);
  applications in Shipping, Warehouse management

2-D Code

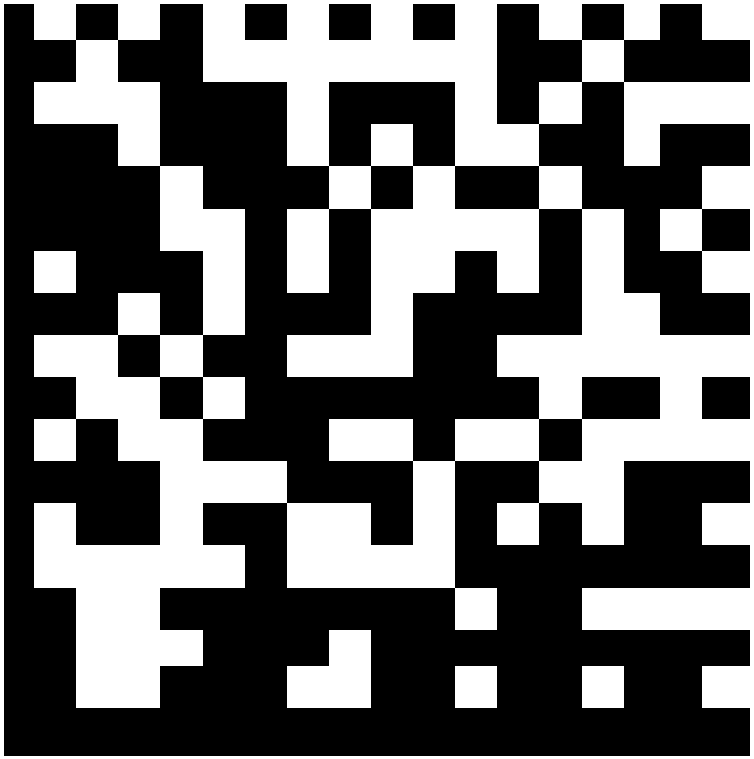
- -PDF417 barcode;

PDF = Portable Data File.

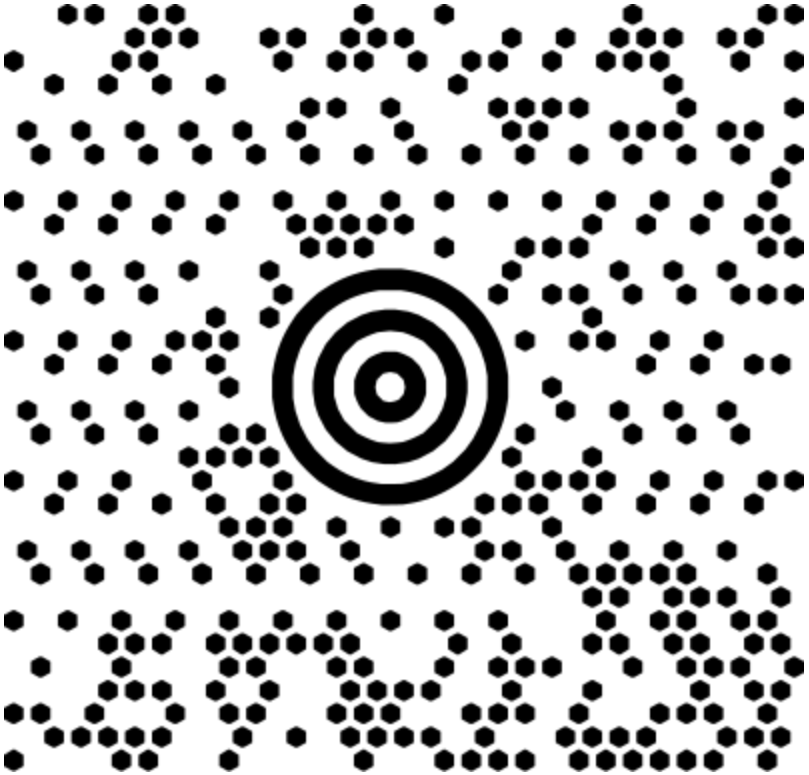
Used in primarily transport, identification cards, and inventory management

https://en.wikipedia.org/wiki/PDF417

More 2D Codes



Data Matrix



MaxiCode

Alphanumeric codes

Alphanumeric codes

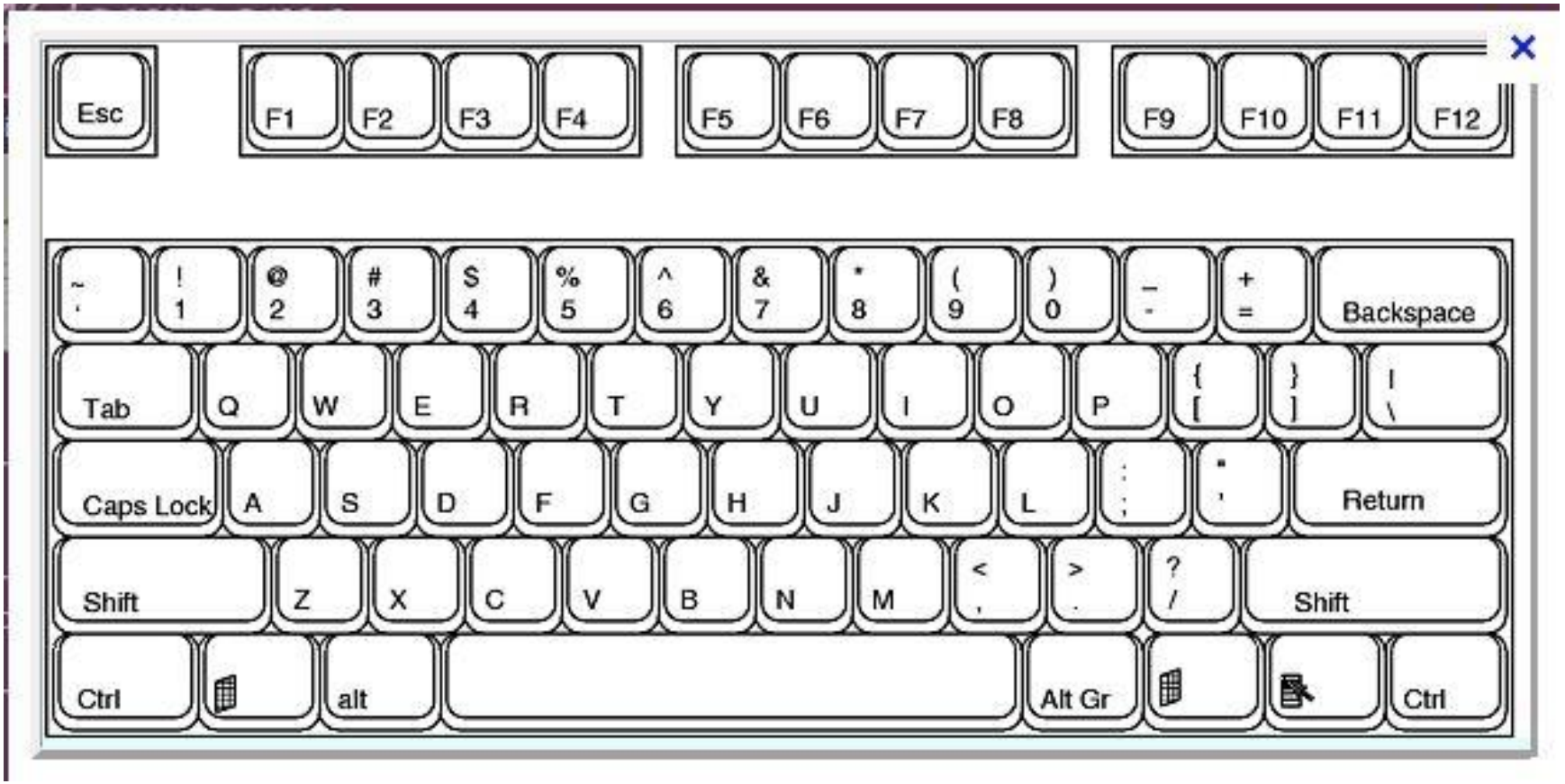
- Used to print, teletype or view information or other means of human alpha-numeric communication.
  - **ASCII**
  - **EBCIDIC**
  - **UniCode**

ASCII

- ASCII
  (American Standard Code for Information Interchange)

7 bits = 128 characters
(now Super ASCII use all 8-bits)

ASCII


Esc
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
~ `
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
_ -
+ =
Backspace
Tab
Q
W
E
R
T
Y
U
I
O
P
{ [
}]
| \
Caps Lock
A
S
D
F
G
H
J
K
L
: ;
" '
Return
Shift
Z
X
C
V
B
N
M
< ,
> .
? /
Shift
Ctrl
alt
Alt Gr
Ctrl

ASCII

ASCII codes for some of the keys on a keyboard.

```
SPACE  =  00100000

A  =  01000001      a  =  01100001
B  =  01000010      b  =  01100010
C  =  01000011      c  =  01100011
D  =  01000100      d  =  01100100
E  =  01000101      e  =  01100101
F  =  01000110      f  =  01100111
```

etc ... etc ...

```
RETURN (end of a line)  =  00001010
```

ASCII = American Standard Code for Information Interchange
(American — that's why '£' isn't a standard character on some printers!)

ASCII

USASCII code chart

b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits b_4 ↓	b_3	b_2	b_1	Row ↓ Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

EBCIDIC

- EBCIDIC
  (Extended BCD Interchange Code)

8 bits = 256 characters

EBCDIC Code Table

B8	B7	B6	B5	B4	B3	B2	B1	HEX-1 \ HEX-0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	NUL	DLE	DS		SP	&	-									0
0	0	0	0	0	0	0	1	1	SOH	SBA	SOS				/		a	j			A	J		1
0	0	0	0	0	0	1	0	2	STX	EUA	FS	SYN					b	k	s		B	K	S	2
0	0	0	0	0	0	1	1	3	ETX	IC							c	l	t		C	L	T	3
0	0	0	0	0	1	0	0	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0	0	0	0	0	1	0	1	5	PT	NL	LF	RS					e	n	v		E	N	V	5
0	0	0	0	0	1	1	0	6	LC		ETB	UC					f	o	w		F	O	W	6
0	0	0	0	0	1	1	1	7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
0	0	0	0	1	0	0	0	8		CAN							h	q	y		H	Q	Y	8
0	0	0	0	1	0	0	1	9		EM							i	r	z		I	R	Z	9
0	0	0	0	1	0	1	0	A	SMM	CC	SM		¢	!	¦	:								
0	0	0	0	1	0	1	1	B	VT				.	$	,	#								
0	0	0	0	1	1	0	0	C	FF	DUP		RA	<	*	%	@								
0	0	0	0	1	1	0	1	D	CR	SF	ENQ	NAK	(	)	_	'								
0	0	0	0	1	1	1	0	E	SO	FM	ACK		+	;	>	=								
0	0	0	0	1	1	1	1	F	SI	ITB	BEL	SUB	|	¬	?	"								

Unicode

- Unicode (standard for 16–bit alphanumeric code)

- Multilanguage (non-Roman) standard

Unicode

	037	038	039	03A	03B	03C	03D	03E	03F
0			ΐ	Π	ΰ	π	ϐ	ϑ	ϰ
1			Α	Ρ	α	ρ	ϑ	ϱ	ϱ
2			Β		β	ς	ϒ	ϲ	ϲ
3			Γ	Σ					
4	ʹ	΄	Δ	Τ					
5	͵	΅	Ε	Υ					
6		Ά	Ζ	Φ					
7		·	Η	Χ					
8		Έ	Θ	Ψ					
9		Ή	Ι	Ω					
A	;	Ί	Κ	Ϊ					
B			Λ	Ϋ					
C		Ό	Μ	ά					
D			Ν	έ					
E	;	Ύ	Ξ	ή					
F		Ώ	Ο	ί					

脚	脡	脢	
8173	8174	8175	
艳	艴	艵	
8273	8274	8275	
苴	苶	苷	
8373	8374	8375	

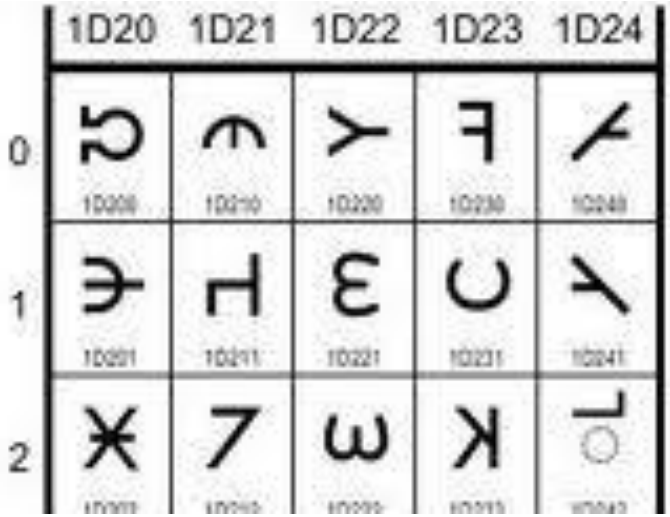
Arial® Unicode® MS Bold

friður 和平 paz שלום ειρήνη 平和

mír शांति мир 평화 frieden শান্তি

WORLDWIDE LANGUAGE SUPPORT

Hòa bình ความสงบสุข Síochána



	1D20	1D21	1D22	1D23	1D24
0	ᴐ	ᴑ	ᴒ	ᴓ	ᴔ
1	ᴕ	ᴖ	ᴗ	ᴘ	ᴙ
2	ᴚ	ᴛ	ᴜ	ᴝ	ᴞ

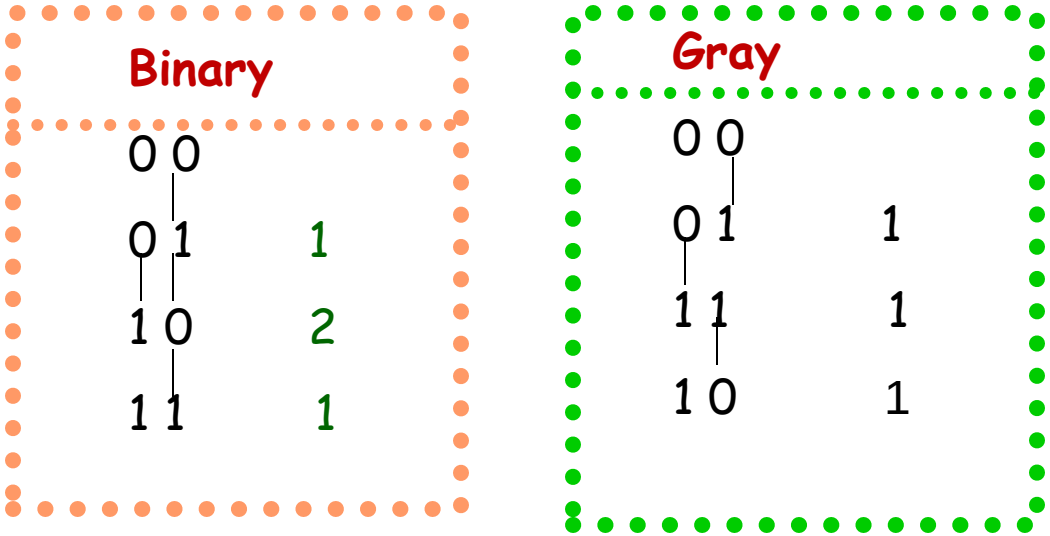
A useful code is the Gray (Frank Gray code)

- Gray code is of cyclic nature having the following property:

“From one number to the next, the Gray code changes only one bit”

Bit changes …

- Gray code is of cyclic nature having the following property:
- “From one number to the next, the Gray code changes only one bit”



Gray Code …

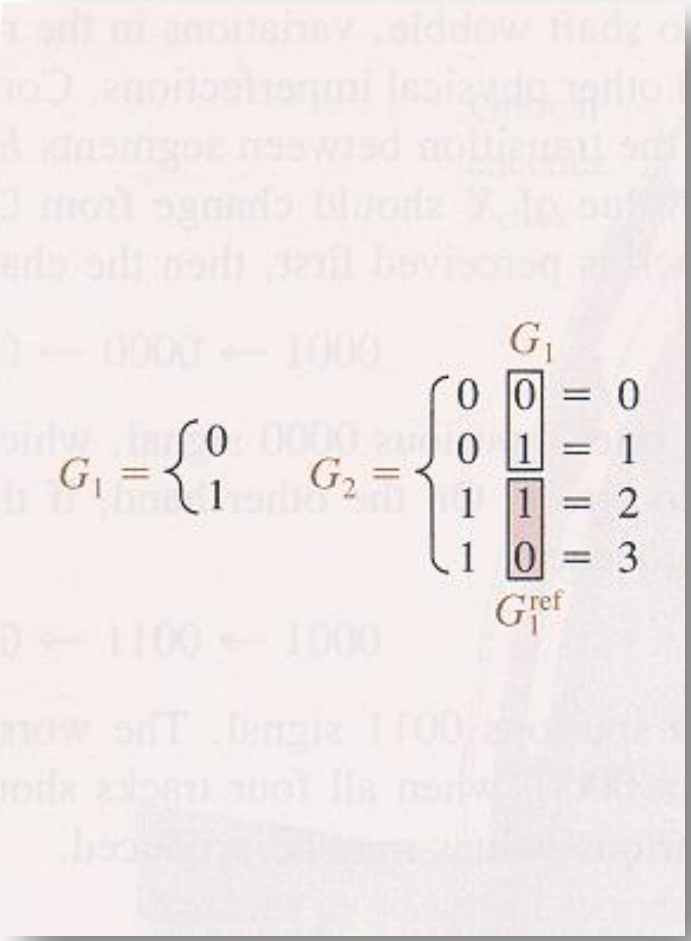
- The Gray code is used for …
- Labeling Karnaugh maps for logic circuit simplification
- Routing in parallel computer systems
- Error correction in modern digital communication systems, etc…

Gray Code = Reflected Binary Code (RBC)

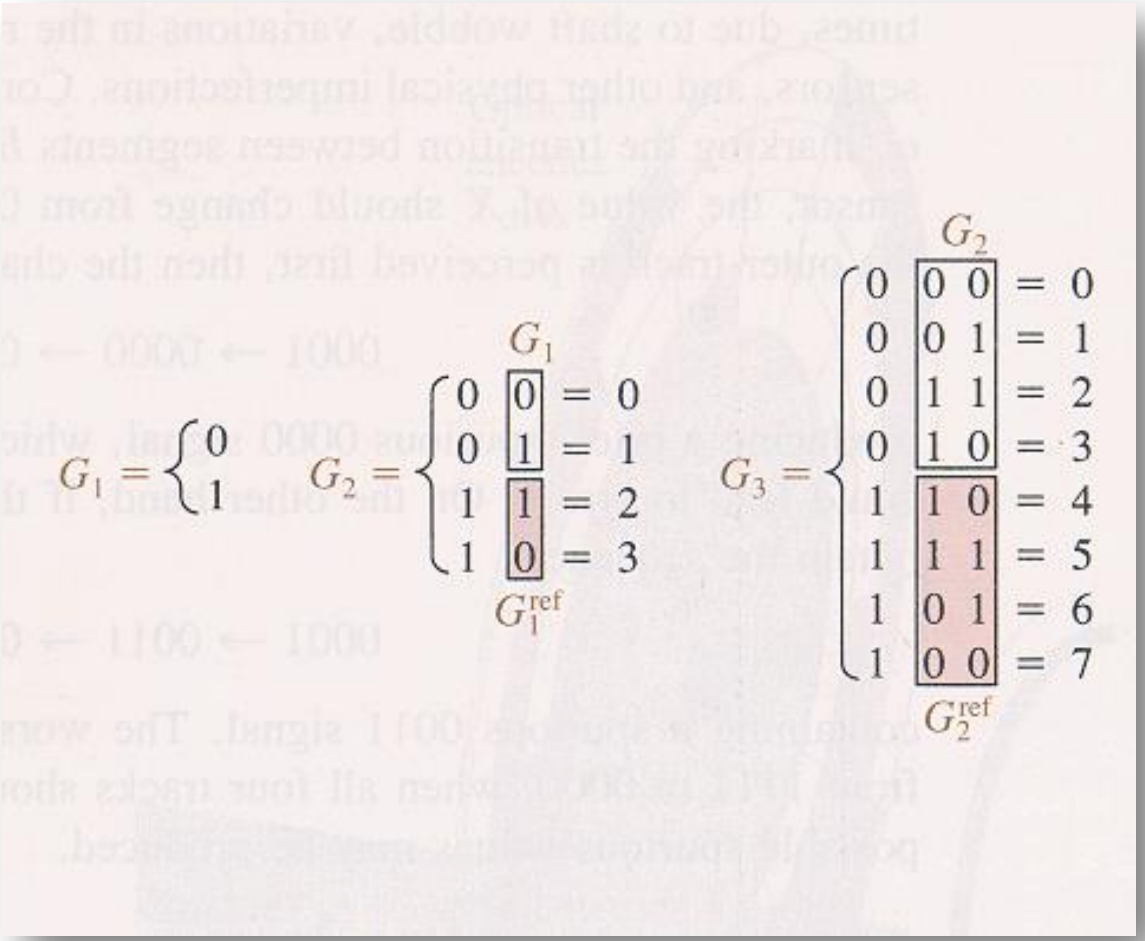
Gray Codes

$$G_1 = \begin{cases} 0 \\ 1 \end{cases}$$

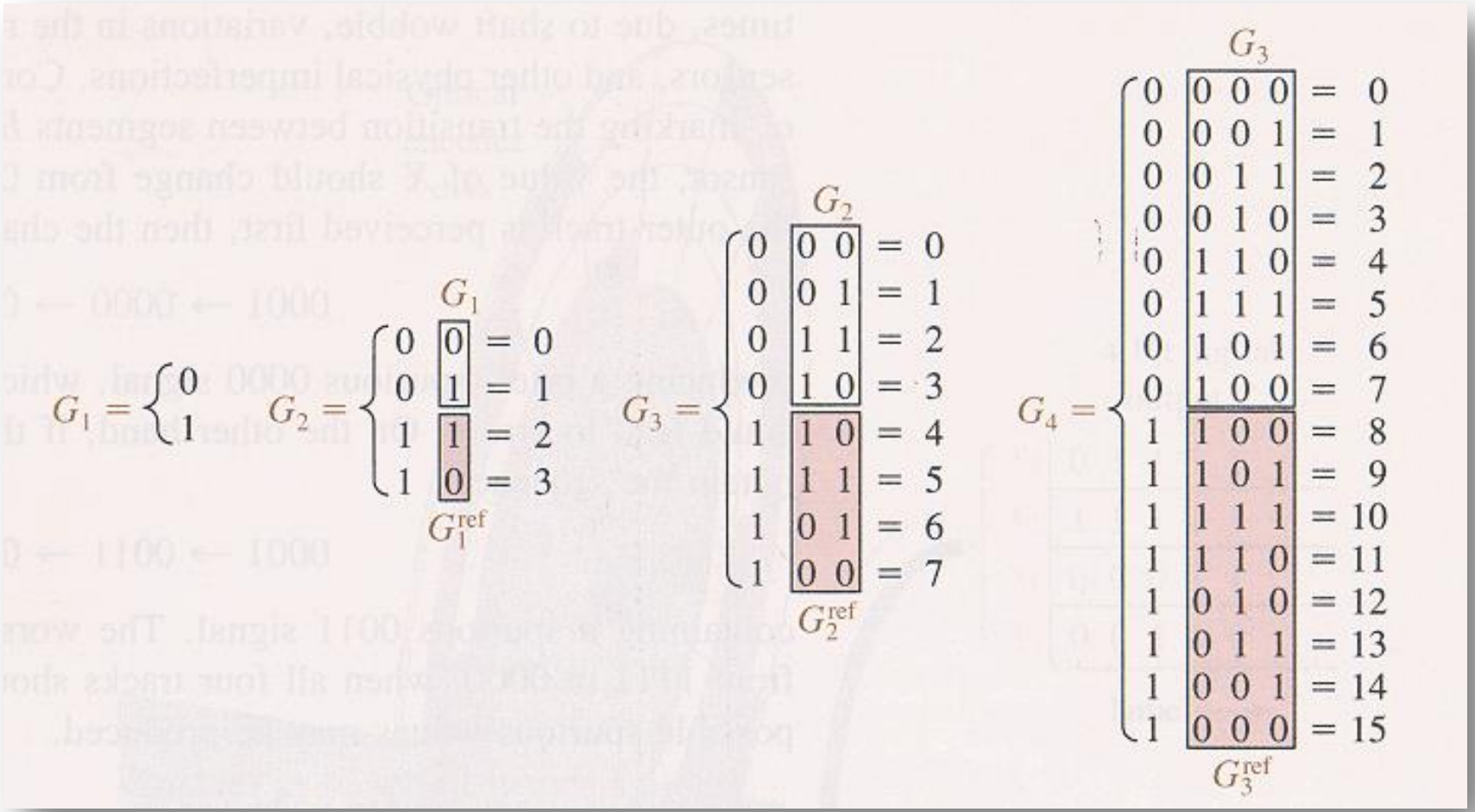
Gray Codes


$G_1 = \begin{cases} 0 \\ 1 \end{cases}$
$G_2 = \begin{cases} 0\ 0 = 0 \\ 0\ 1 = 1 \\ 1\ 1 = 2 \\ 1\ 0 = 3 \end{cases}$
G_1
G_1^{ref}

Gray Codes


$G_1 = \begin{cases} 0 \\ 1 \end{cases}$
$G_2 = \begin{cases} 0\ \boxed{0} = 0 \\ 0\ \boxed{1} = 1 \\ 1\ \boxed{1} = 2 \\ 1\ \boxed{0} = 3 \end{cases}$
G_1
G_1^{ref}
$G_3 = \begin{cases} 0\ \boxed{0\ 0} = 0 \\ 0\ \boxed{0\ 1} = 1 \\ 0\ \boxed{1\ 1} = 2 \\ 0\ \boxed{1\ 0} = 3 \\ 1\ \boxed{1\ 0} = 4 \\ 1\ \boxed{1\ 1} = 5 \\ 1\ \boxed{0\ 1} = 6 \\ 1\ \boxed{0\ 0} = 7 \end{cases}$
G_2
G_2^{ref}

Gray Codes


$G_1 = \begin{cases} 0 \\ 1 \end{cases}$
$G_2 = \begin{cases} 0\,0 = 0 \\ 0\,1 = 1 \\ 1\,1 = 2 \\ 1\,0 = 3 \end{cases}$
G_1
G_1^{ref}
$G_3 = \begin{cases} 0\,0\,0 = 0 \\ 0\,0\,1 = 1 \\ 0\,1\,1 = 2 \\ 0\,1\,0 = 3 \\ 1\,1\,0 = 4 \\ 1\,1\,1 = 5 \\ 1\,0\,1 = 6 \\ 1\,0\,0 = 7 \end{cases}$
G_2
G_2^{ref}
$G_4 = \begin{cases} 0\,0\,0\,0 = 0 \\ 0\,0\,0\,1 = 1 \\ 0\,0\,1\,1 = 2 \\ 0\,0\,1\,0 = 3 \\ 0\,1\,1\,0 = 4 \\ 0\,1\,1\,1 = 5 \\ 0\,1\,0\,1 = 6 \\ 0\,1\,0\,0 = 7 \\ 1\,1\,0\,0 = 8 \\ 1\,1\,0\,1 = 9 \\ 1\,1\,1\,1 = 10 \\ 1\,1\,1\,0 = 11 \\ 1\,0\,1\,0 = 12 \\ 1\,0\,1\,1 = 13 \\ 1\,0\,0\,1 = 14 \\ 1\,0\,0\,0 = 15 \end{cases}$
G_3
G_3^{ref}

Binary and BCD numbers

Binary and BCD numbers

- Bits obtained from conversion, of binary numbers, are binary digits
- Bits obtained from coding (BCD) are combinations of 1’s and 0’s arranged according to the rules of the code (BCD).

Example-1

- $(13)_{10} = (1101)_2$

Example-1

- $(13)_{10} = (1101)_2$

but,

$(13)_{10} = (0001\ \ 0011)_{BCD}$

$(1\ \ \ \ \ \ 3)_{10}$

Example-2

- $(7)_{10} = (111)_2$

Example-2

- $(7)_{10} = (111)_{2}$

but,,

$(7)_{10} = (0111)_{BCD}$

Another Example

- $(15)_{10} = (1111)_2$

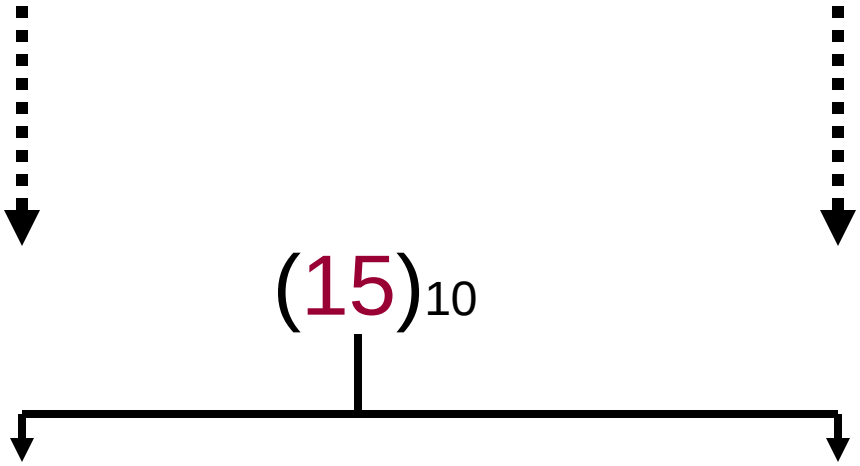
Another Example

- $(15)_{10} = (1111)_2$

but,

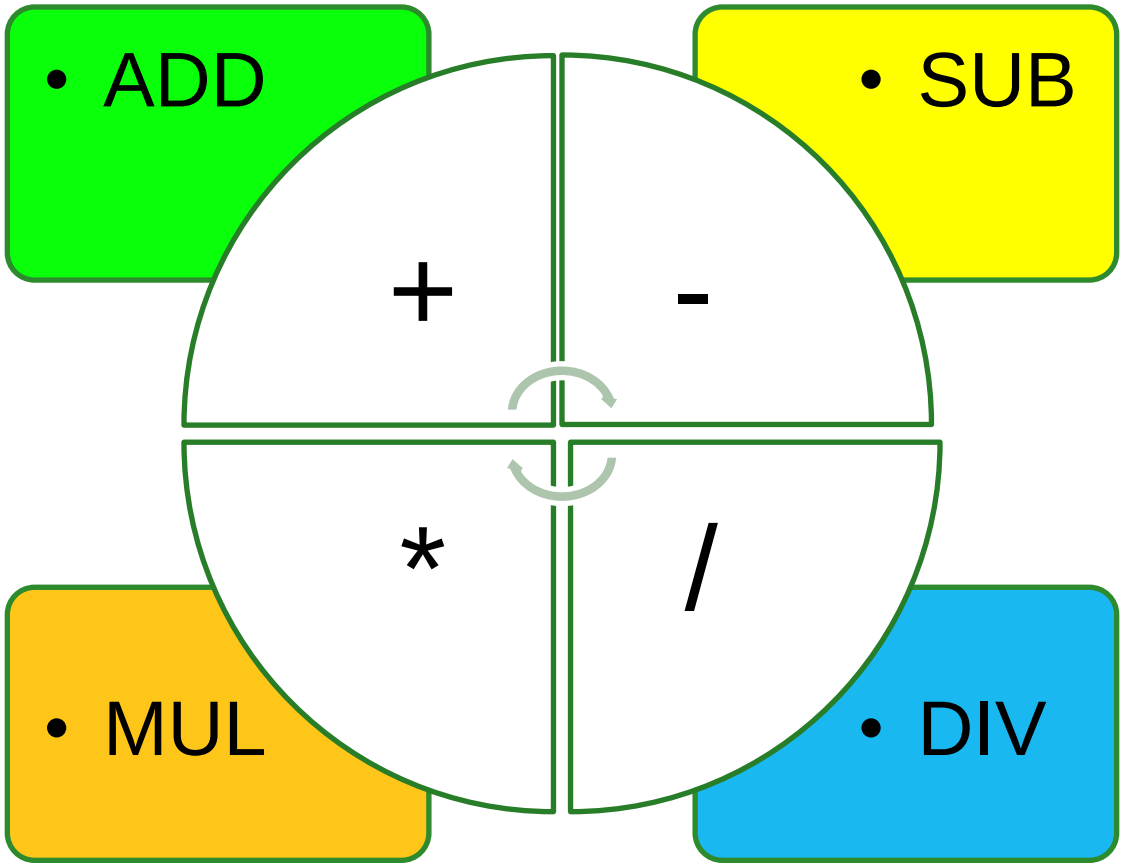
$(15)_{10} = (0001\ \ 0101)_{BCD}$

Binary & BCD



$(1111)_2$ $(0001\ \ 0101)_{BCD}$

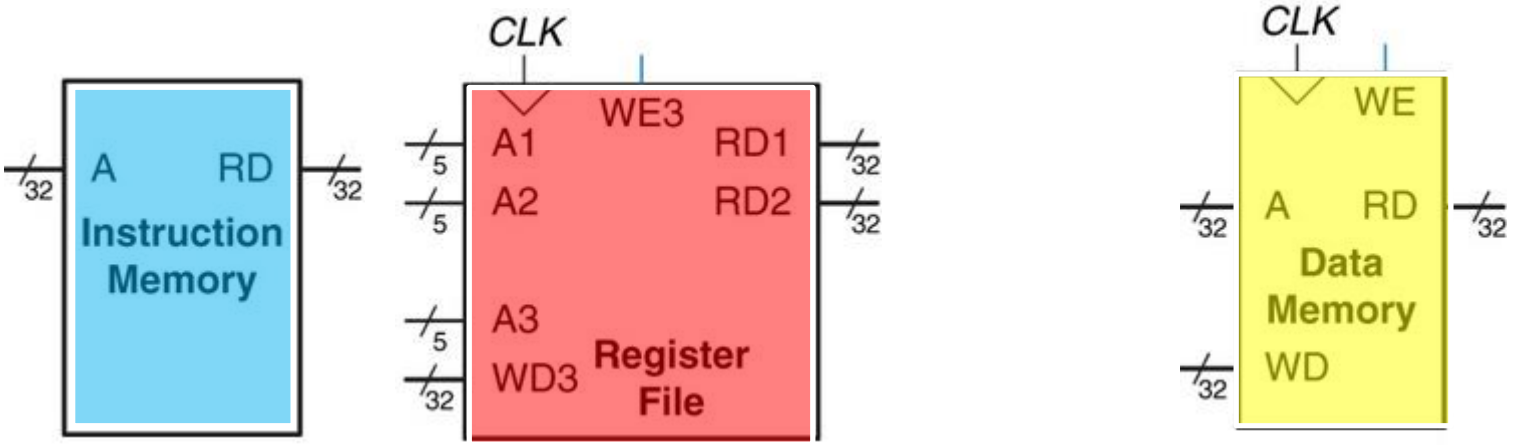
Arithmetic operations for humans



Computer System

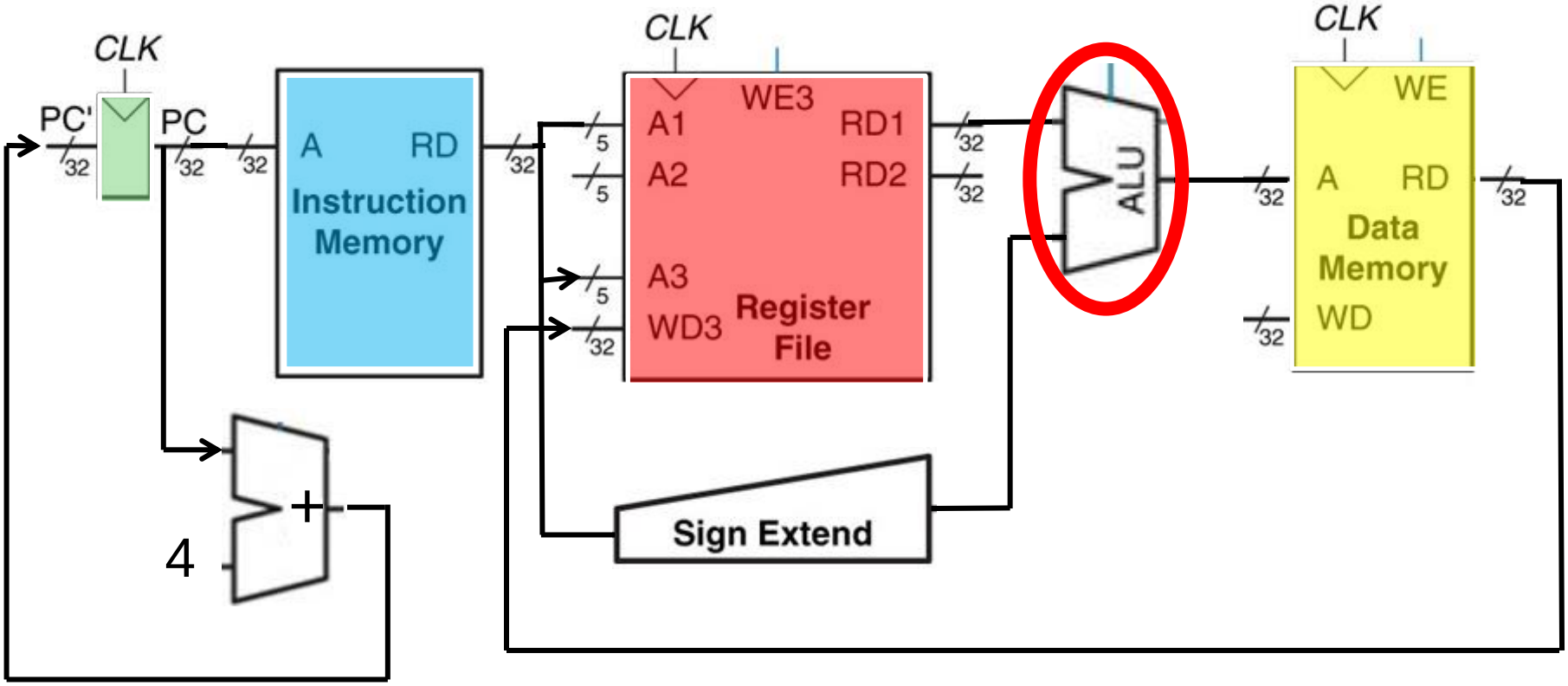
Microarchitecture

CPU … IM, DM


A
RD
Instruction Memory
32
32
CLK
WE3
A1
A2
A3
WD3
RD1
RD2
Register File
5
5
5
32
32
32
CLK
WE
A
RD
Data Memory
WD
32
32
32

CPU … and ALU

The ALU (Arithmetic and Logic Unit) performs: Arithmetic and Logic operations



ALU is the heart of the CPU

- The computer-Central Processing Unit (CPU) performs logical and arithmetic operations
- Main arithmetic operation in the Arithmetic Logic Unit (ALU) of the CPU?
  - **ADD (addition)**
- All the other arithmetic operations are performed, using addition, and …
- … how ?

Addition

Decimal Addition

3	6	9
+	3	2

Decimal Addition

	1	
3	6	9
+	3	2
		1

Decimal Addition

1	1		Carry
3	6	9	
+	3	2	
4	0	1	SUM

Binary Addition

Binary Addition

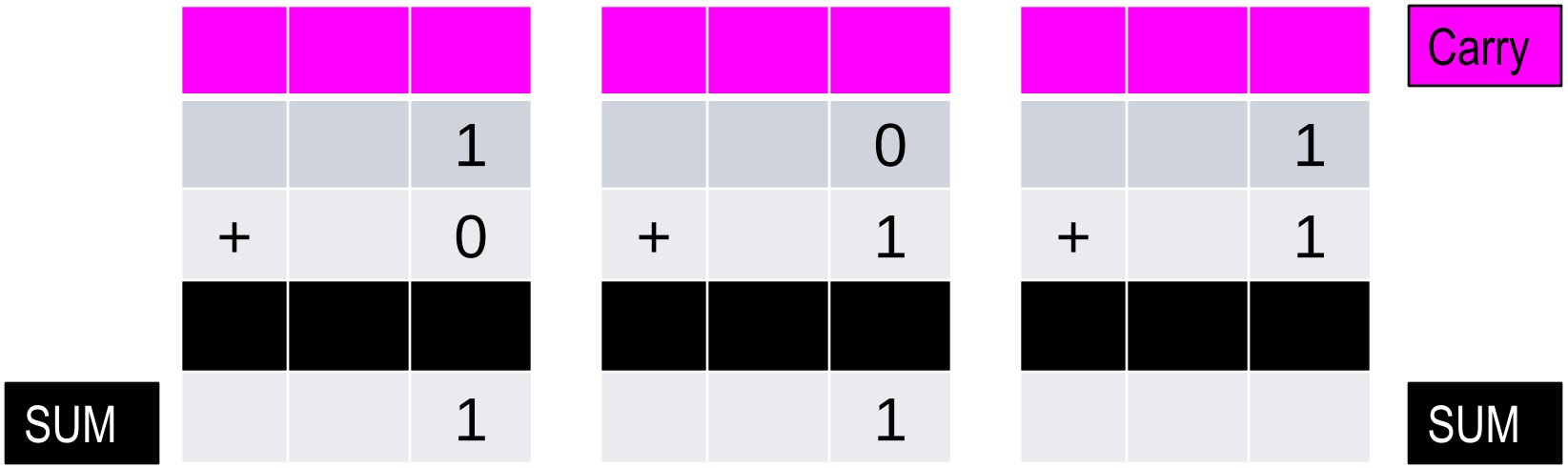
			1
	+		0
SUM			1

Binary Addition

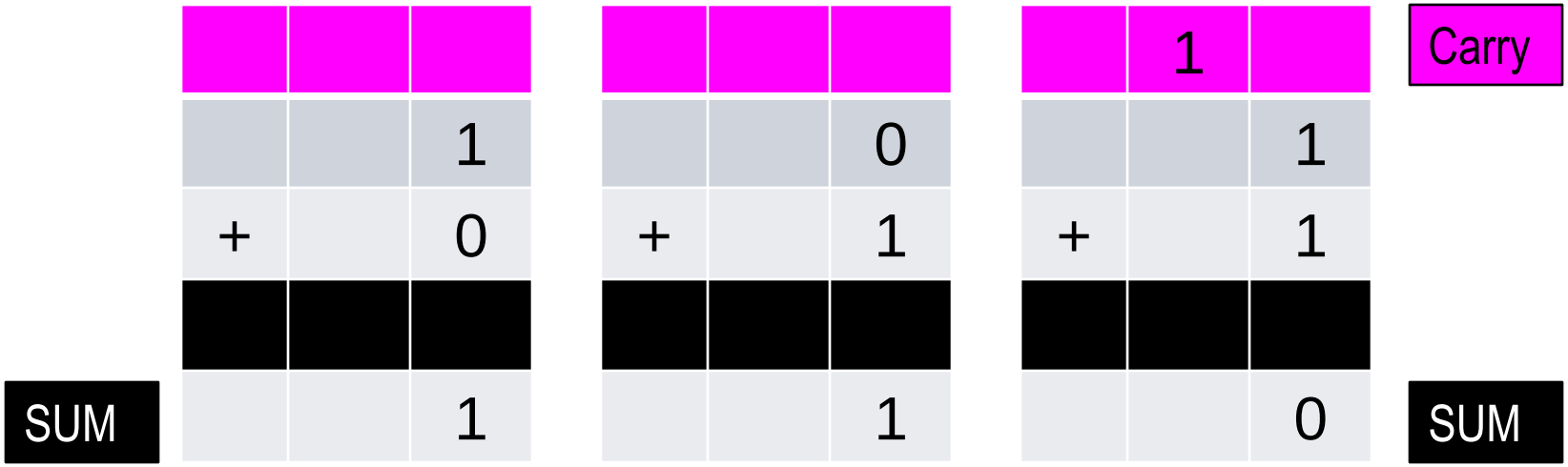
		1	
	+	0	
SUM		1	

		0	
	+	1	
		1	

Binary Addition



Binary Addition



Binary example

Binary example

10110

+10011

_____

1 ← Result

Binary example

0 1 1 0 0 ← Carry

10110

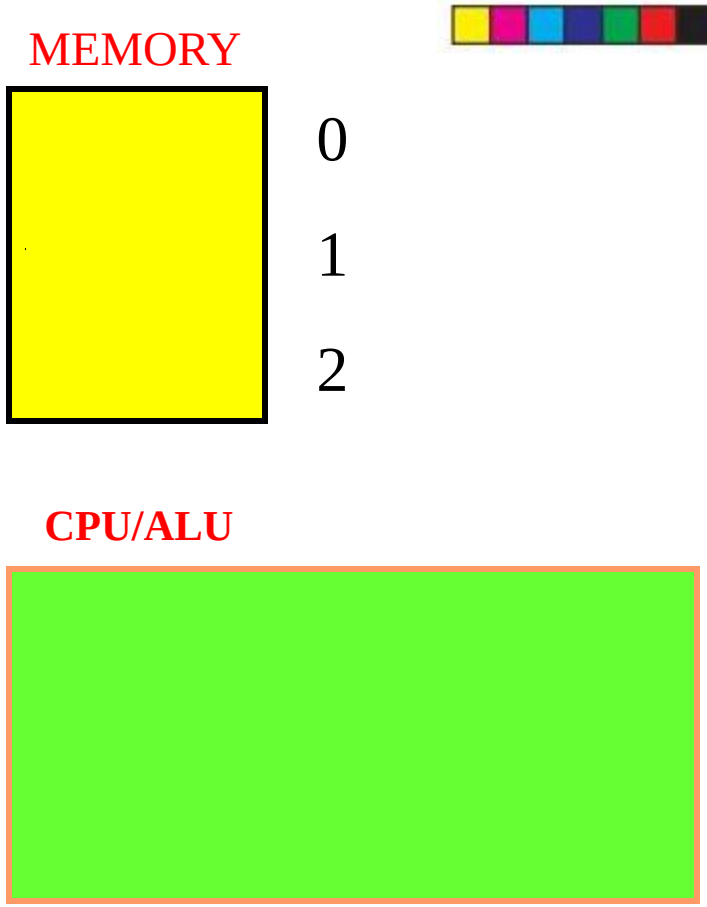
+10011

101001 ← Result

MSBit LSBit

To add two binary numbers … let us use an ALU and Memory…

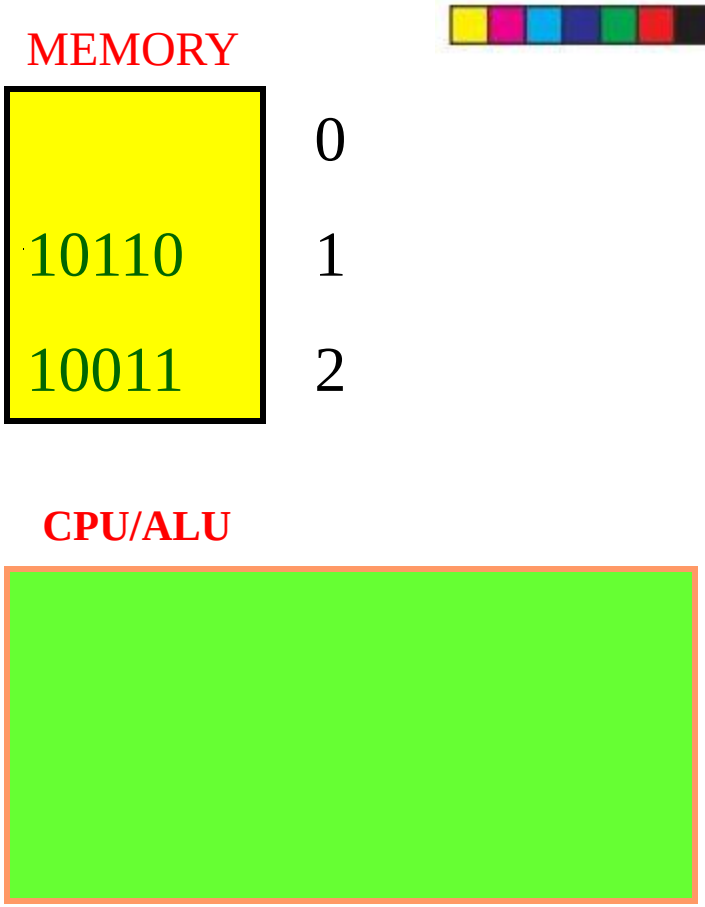
Addition using the ALU (CPU)-MEMORY



STORE both numbers in the memory…

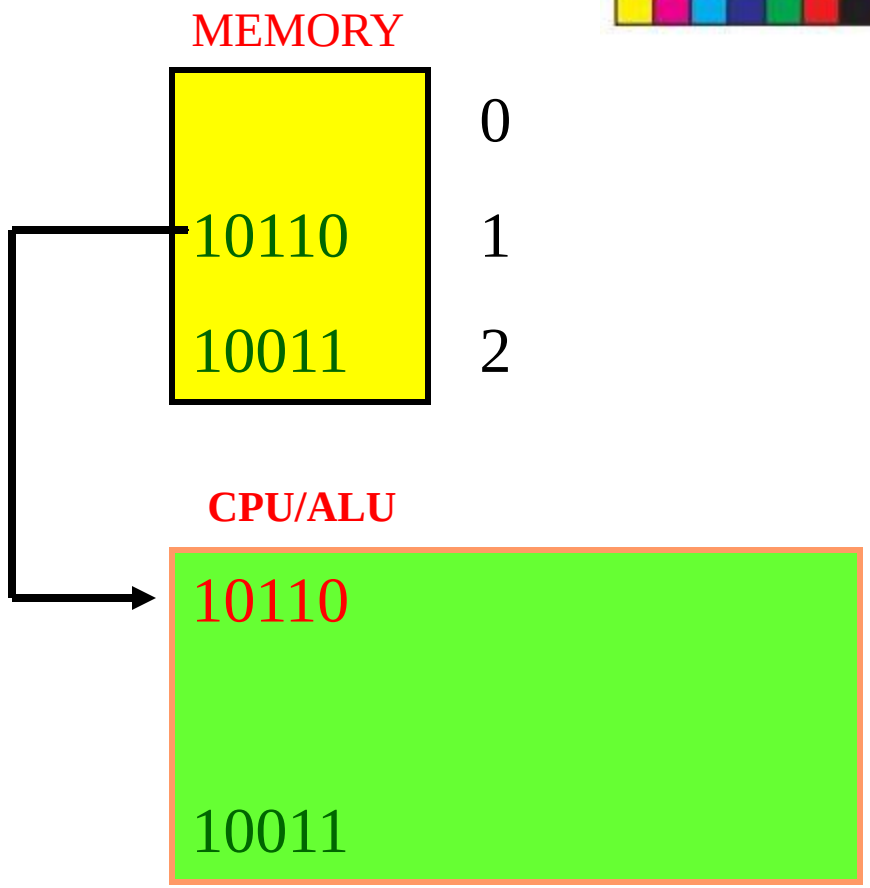
Step-1:Store in the memory: 10110

Step-2:Store in the memory: 10011



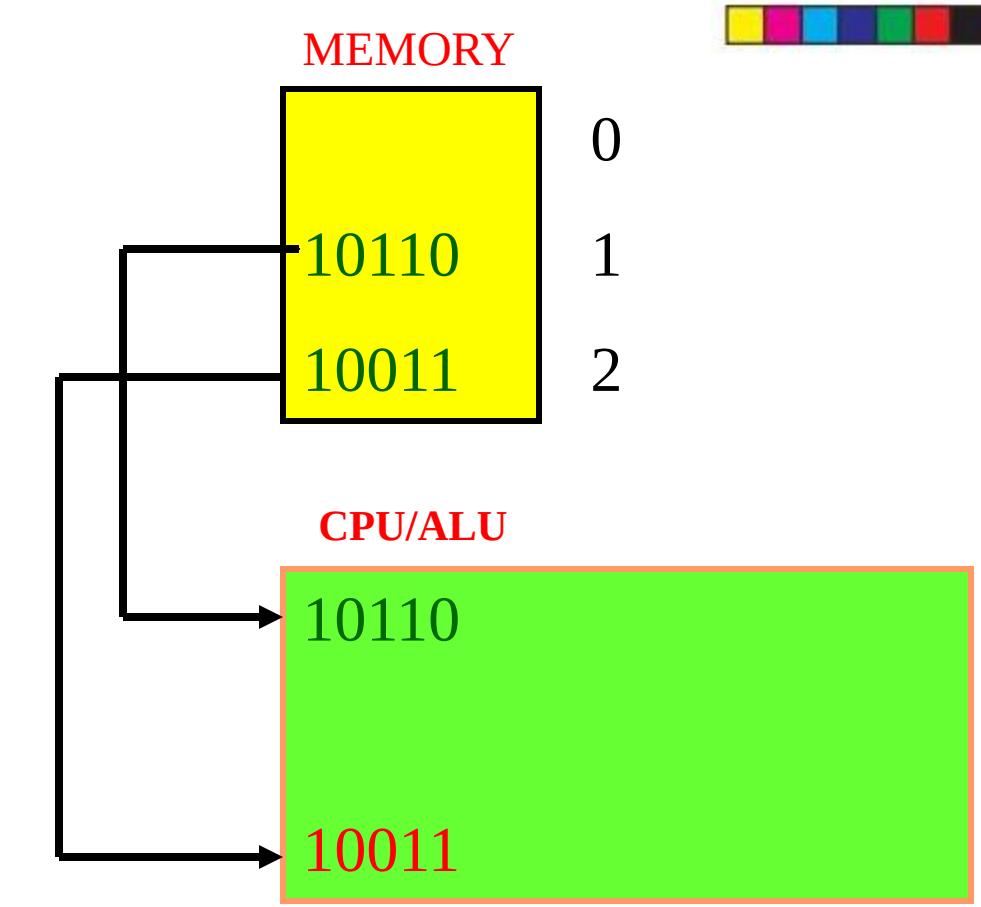
LOAD first number to ALU…

Step-3: Load 10110 to ALU

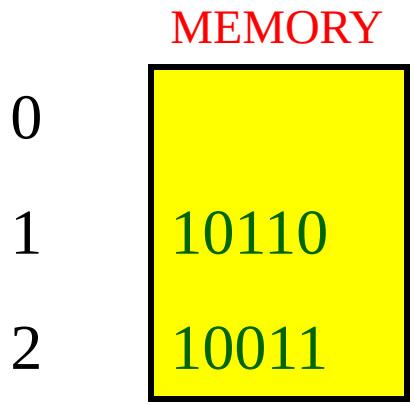


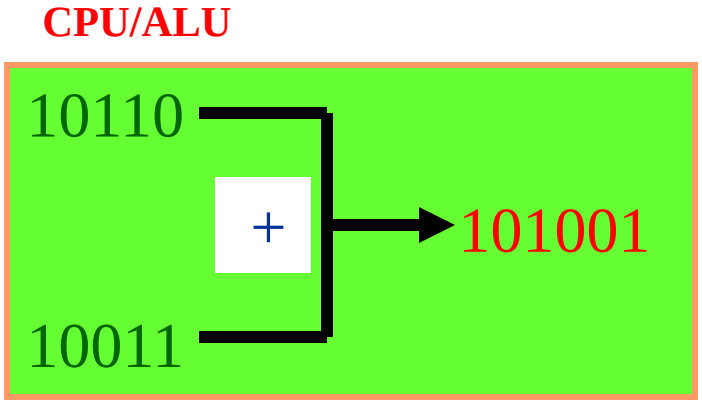
LOAD second number to ALU…

Step-4: Load 10011 to ALU



ADD using the ALU (CPU)

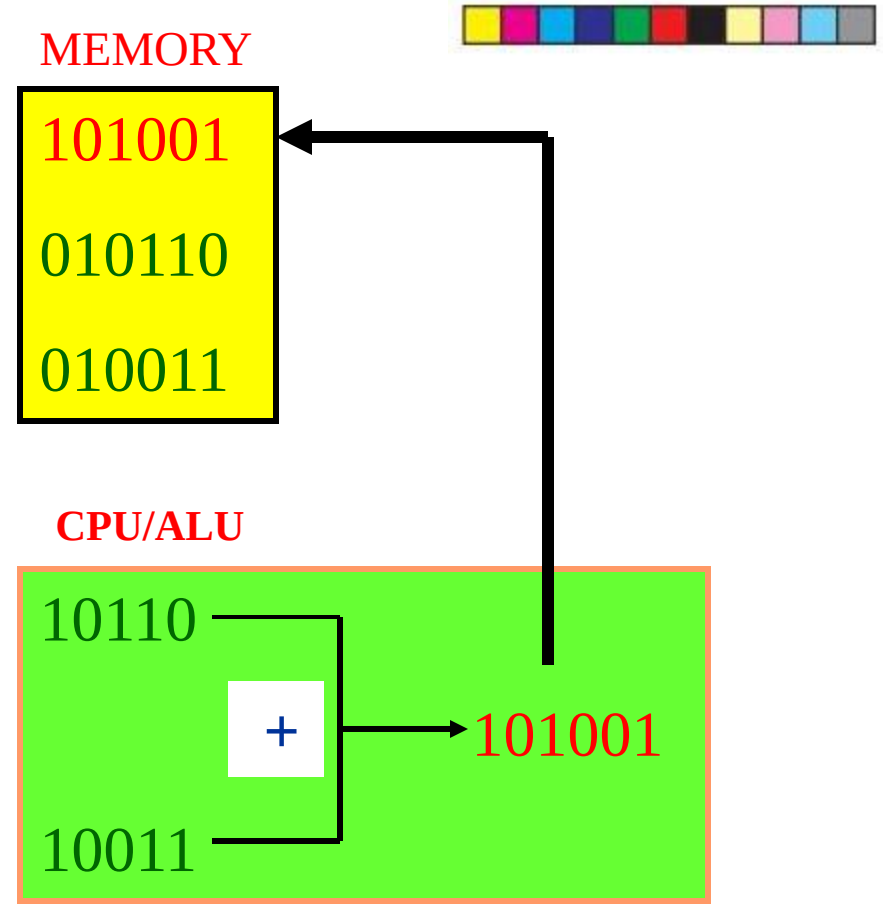




Step-5: Add 10110+10011

STORE the result back to Memory

Step-6: Store 01001 to Memory



10011, 10110 and 101001 are stored in the CPU/ALU in Registers

Addition using octal and hex

Addition using Octal (1)

- 1476
- +3554

  ????

Addition using Octal (2)

- 1476 6+4 = 10 => (1, 2)
- +3554 carry sum

1

1476

3554

2

Addition using Octal (3)

- 1476 1+4+5 = 10 => (1, 2)
- +3554 carry sum

111

1476

3554

252

Addition using Octal (4)

- 1476 1+1+3 = 5 => (0, 5)
- +3554 carry sum

111

1476

3554

5252

Addition using Hex (1)

- 59F
- +E46

???

Addition using Hex (2)

- 59F F + 6 = 15 + 6 => (1, 5)
- +E46 carry sum

```
  1
 59F
 E46
-----
   5
```

Addition using Hex (3)

- 59F 1 + 9 + 4 = 14 => (0, E)
- +E46 carry sum

```
  01
  59F
  E46
 ----
  E5
```

Addition using Hex (4)

- 59F 0 + 5 + E = 19 => (1, 3)
- +E46 carry sum

01

59F

E46

13E5

Multiplication

Binary Multiplication

101

X 100

000	100 add zeros
000	100 shift-add zeros
+101	100 shift-copy number
10100	ADD

Binary Multiplier

- We need two hardware components:
  - Shifter
  - Adder

Division

Binary Division

- We need two hardware components:
  - Shifter
  - Subtract

Subtraction

Decimal subtraction

4	0	1
-	3	2

Decimal subtraction

1	1		Borrow
4	0	1	
-	3	2	
3	6	9	result

Binary Subtraction

Binary subtraction

		0
-		0
		0

Binary subtraction

		0
-		0
		0

		1
-		0
		1

Binary subtraction

		0
-		0
		0

		1
-		0
		1

		1
-		1
		0

Binary subtraction

										1		Borrow
		0			1			1			0	
-		0	-		0	-		1	-		1	
		0			1			0			1	result

Subtract: 10110-10011

1 0 1 1 0 = 2+4+16 = 22

- 1 0 0 1 1 = 1+2+16 = 19

---

(1)

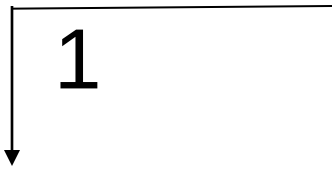
Since 0-1

1 0 1 1 0

- 1 0 0 1 1

(a)

Borrow 1 from next (left) column



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

(b)

Borrow 1 from next (left) column

```
      1
      ↗
  1   0   1   1   0
- 1   0   0   1   1
-----------------------
                  1
                  ↑
```

Therefore 10-1 = 1

(c)

Since we borrowed a 1 a 0 is left

			0	1	
	1	0	1	~~1~~	0
-	1	0	0	1	1
					1

(d)

Since 0 – 1, borrow 1 from next (left) column

0 1

1 0 1 1 0

- 1 0 0 1 1

1

(e)

Borrow 1 from next (left) column

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

(f)

Borrow 1 from next (left) column

1 0 1

1 0 1 ~~1~~ 0

- 1 0 0 1 1

---

1 1

Therefore 10-1 = 1

(g)

Since we borrowed a 1 a 0 is left

$$
\begin{array}{r}
 & & 0 & & \\
\hline
1 & 0 & \cancel{1} & 1 & 0 \\
-\,1 & 0 & 0 & 1 & 1 \\
\hline
 & & & 1 & 1 \\
 & & & \uparrow &
\end{array}
$$

(h)

Since we borrowed a 1 a 0 is left

			0		
	1	0	~~1~~	1	0
-	1	0	0	1	1
			0	1	1

Therefore 0-0 = 0

(i)

1 0 1 1 0

- 1 0 0 1 1

---

0 0 1 1

0-0 = 0

(j)

1 0 1 1 0

- 1 0 0 1 1

---

0 0 0 1 1

1-1 = 0

Result: 22-19=3

$$
\begin{array}{rrrrr}
1 & 0 & 1 & 1 & 0 \\
-\,1 & 0 & 0 & 1 & 1 \\
\hline
0 & 0 & 0 & 1 & 1
\end{array}
$$

Therefore

- The classic binary subtraction is somewhat lengthy and complicated to implement …
- … to get around this …
- The modern computer technology uses …

- … Complements

Complements

- In computing the following complements are used:
  - 1’s complement (binary)
  - 2’s complement (binary).

https://en.wikipedia.org/wiki/Two%27s_complement

1’s complement-binary numbers

- Replace … all zero’s with one’s, and the one’s with zero’s

- Example:
  - Binary number: 100001
  - 1’s complement: 011110

2’s complement-binary numbers

- If the binary number is already in 1’s complement … then…
- Add 1 …
- … the result is the 2’s complement of the initial binary number
- Example:
- Binary number: 100001
- 1’s complement of 100001= 011110

+ 1

011111 **2’s complement**

Let’s subtract using 2’s complement

$$x - y$$

(M-N) algorithm using (2's complement)

1. Find 2's complement of the negative number Y

2. Add X to 2's complement of Y
   a) If an end carry occurs, discard it and whatever is left is your answer
   b) If an end carry does not occur, take the 2's complement of the number obtained in step 1 and place a minus (-) sign in front of it. (The result is a negative number in 2's complement)

Example 1

1) Sub …1010100-1000100, using 2's comp.

- X = 101 0100 = 84
- -Y = -100 0100 = -68

16

Step-1: Find the 2's complement of the negative number (-68)

1) Sub …1010100-1000100, using 2's comp.

- X = 101 0100 = 84
- -Y = -100 0100 = -68

16

2's of 100 0100 ?

1) Sub …1010100-1000100, using 2's comp.

- X = 101 0100 = 84
- -Y = -100 0100 = -68

16

2's of 100 0100 ?

1's = 011 1011

+ 1

1) Sub …1010100-1000100, using 2's comp.

- X = 101 0100 = 84
- -Y = -100 0100 = -68

16

2's of 100 0100 ?

1's = 011 1011

+ 1

011 1100

Step-2: ADD the 2's complement of the negative number (-68) with the positive number (84)

1) Sub …1010100-1000100, using 2's comp.

- X = 101 0100 = 84
- -Y = -100 0100 = -68

16

Therefore,

2's of 100 0100 ?

1's = 011 1011

+ 1

011 1100

101 0100

\+ 011 1100

1 001 0000

1) Sub …1010100-1000100, using 2's comp.

- X = 101 0100 = 84
- -Y = -100 0100 = -68

16

Therefore,

2's of 100 0100 ?

1's = 011 1011

+ 1

011 1100

101 0100

+ 011 1100

1 001 0000

The result is: 001 0000 = $(16)_{10}$

The answer

Therefore the subtraction of binary numbers, using complements,…

… is equivalent to addition

Example 2

2) Sub …1000100-1010100, using 2's comp.

- X = 100 0100 = 68
- -Y = -101 0100 = -84

-16

Step-1: Find the 2's complement of the negative number (-84)

Sub …1000100-1010100, using 2's comp.

- X = 100 0100 = 68
- -Y = -101 0100 = -84

-16

2's of 1010100 ?

1's = 0101011

+ 1

010 1100

Step-2: ADD the 2's complement of the negative number (-84) with the positive number (68)

Sub …1000100-1010100, using 2's comp.

- X = 100 0100 = 68
- -Y = -101 0100 = -84

-16

Therefore,

2's of 1010100 ?

1's = 0101011

+ 1

010 1100

100 0100

+ 010 1100

NC 111 0000

If there is no Carry-out the result is NEGATIVE and in 2's Complement form

Sub …1000100-1010100, using 2's comp.

- X = 100 0100 = 68
- -Y = -101 0100 = -84

-16

Therefore,

2's of 1010100 ?

1's = 0101011

+ 1

010 1100

100 0100

+ 010 1100

NC 111 0000

Since we use unsigned arithmetic …place a negative sign in front of the number

The result is: - 2's of (111 0000) = -10000 = $(-16)_{10}$

Negative numbers

- Our result was (binary) negative [-10000]. Since the minus sign is not a binary symbol …
- … we need a way to represent negative and positive numbers using the binary symbols: (1, 0)
- For this reason the **signed binary** number representation will be introduced.

Signed binary numbers (Next Lecture)

1. Unsigned integers in binary

2. Representing **signed** binary integers:
   a) Sign-magnitude
   b) One's complement signed magnitude
   c) Two's complement signed magnitude

3. Representing binary integers/fractions:
   a) Fixed-point numbers
   b) Floating-point numbers