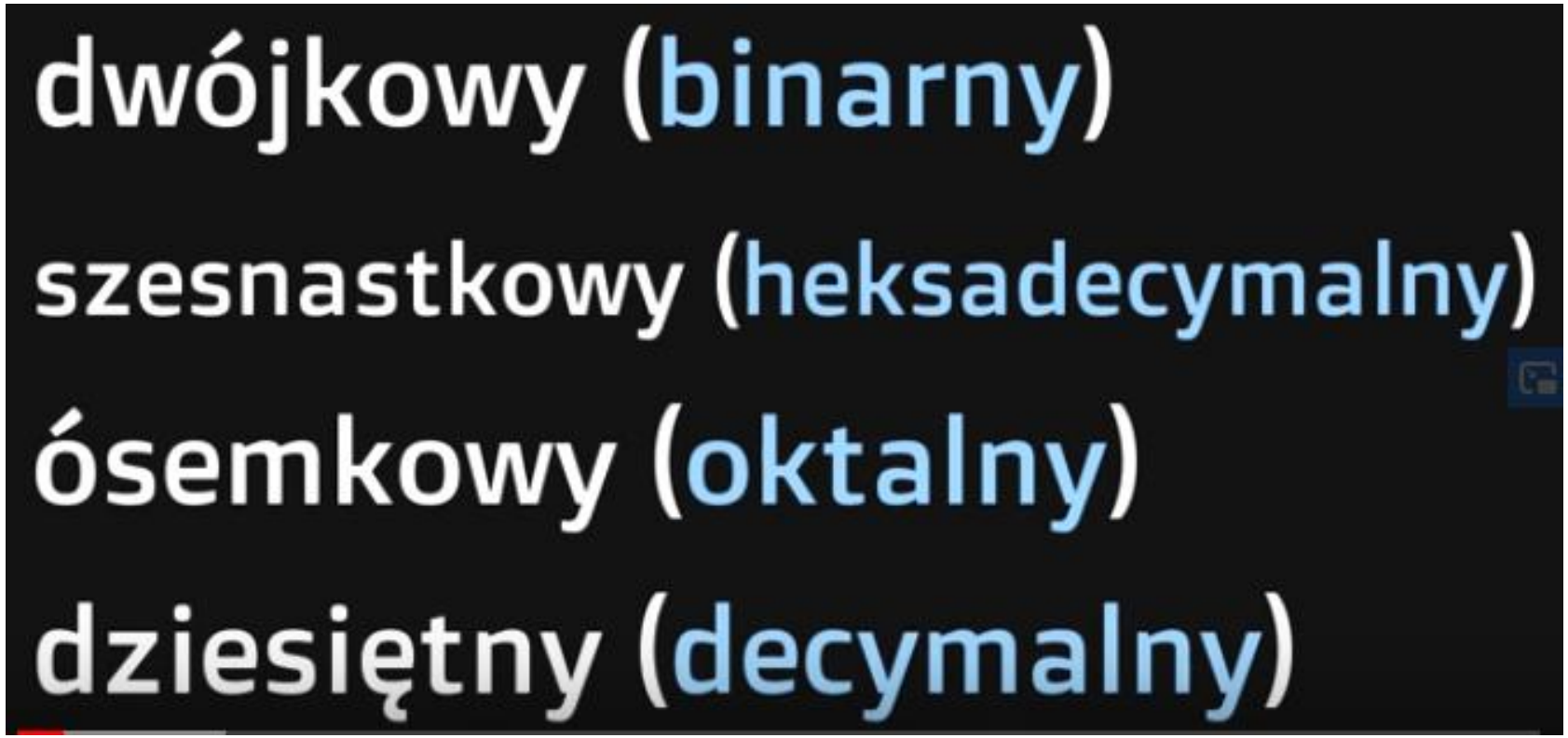
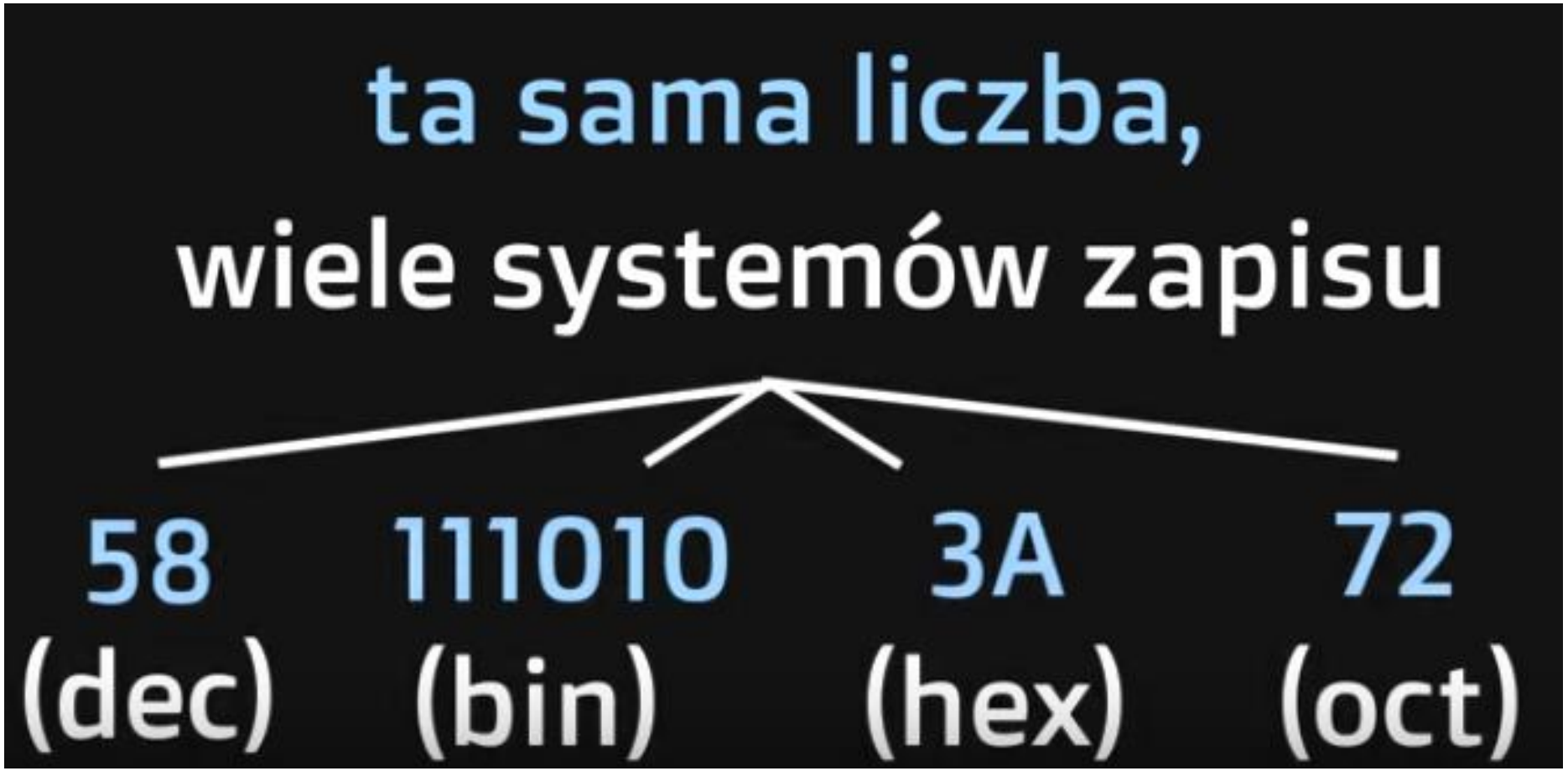
Systemy liczbowe arytmetyka

Reprezentacja liczb



Reprezentacja liczb



Reprezentacja liczb

Majowie zamieszkiwali południowo-wschodnią część Meksyku, Gwatemalę i część Hondurasu. Wprowadzili pojęcie zera, opracowali pozycyjny system liczbowy przed wprowadzeniem symboli arabskich na terytorium Europy oraz własny kalendarz.

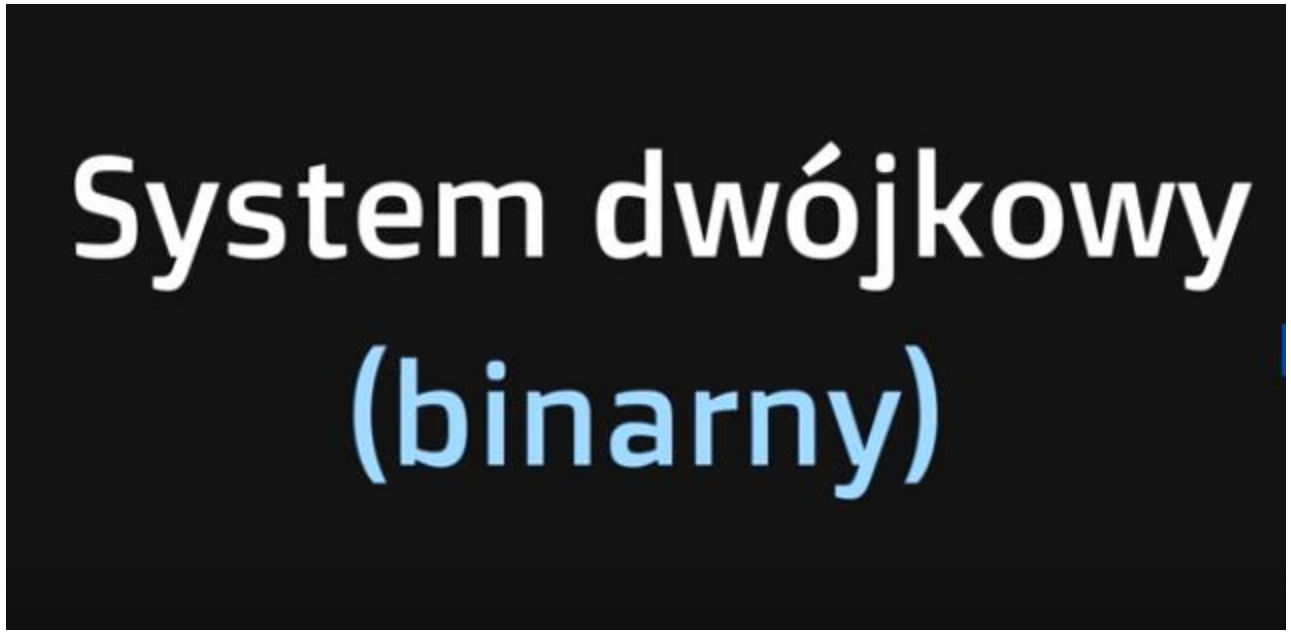
Posługiwali się systemem dwudziestkowym, który opierał się na trzech symbolach: kropka, kreska i muszla. Znak kropki oznaczał jednostkę. Pozioma kreska oznaczała piątkę. Muszla zero.

Reprezentacja liczb

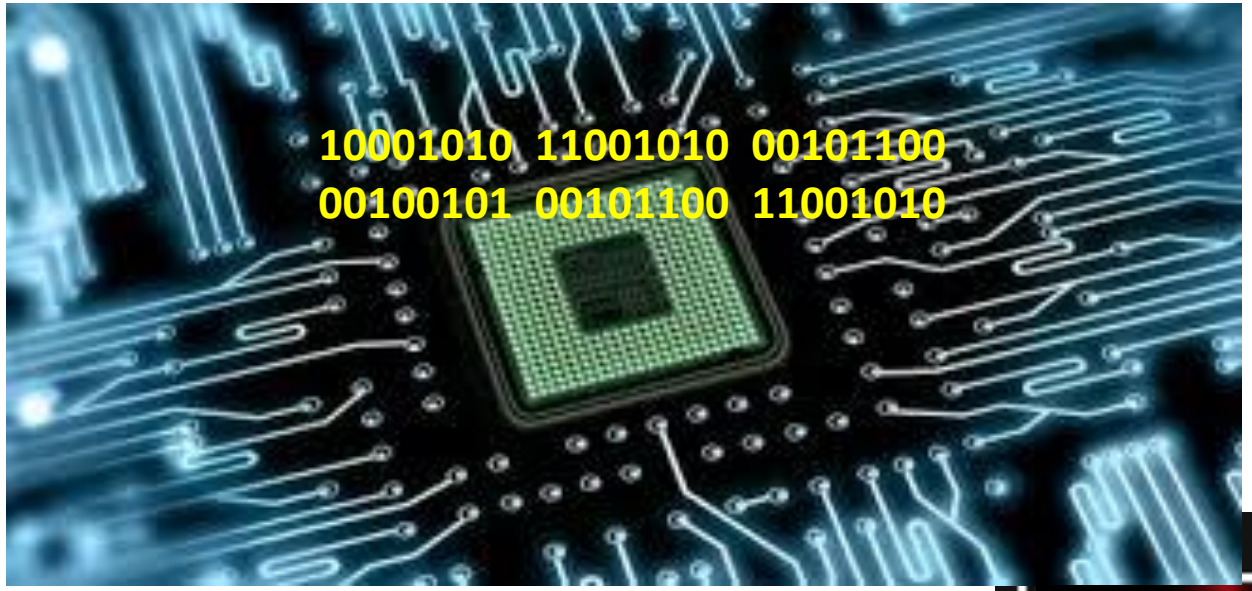
Ilość znaków w systemach

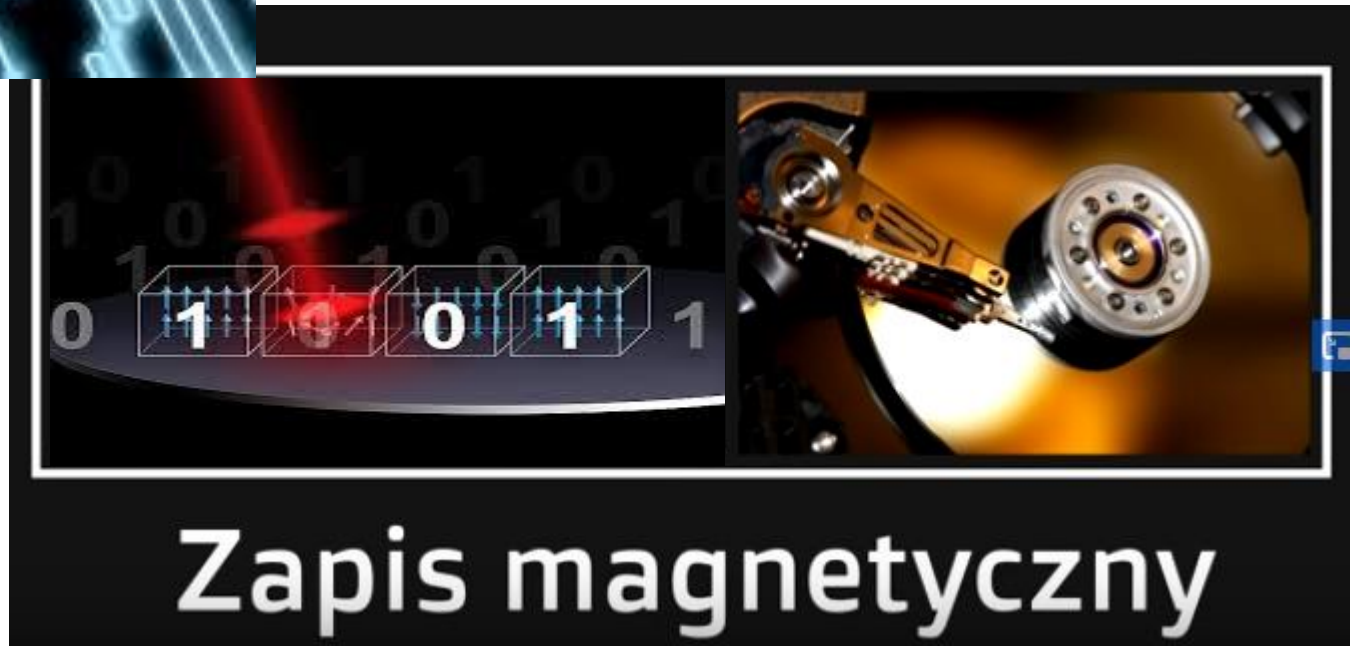
dwójkowy	ósemkowy	dziesiętny	szesnastkowy
2 symbole:	8 symboli:	10 symboli:	16 symboli:
0	0 4	0 5	0 5 A (=10)
1	1 5	1 6	1 6 B (=11)
	2 6	2 7	2 7 C (=12)
	3 7	3 8	3 8 D (=13)
		4 9	4 9 E (=14)
			F (=15)

Reprezentacja liczb

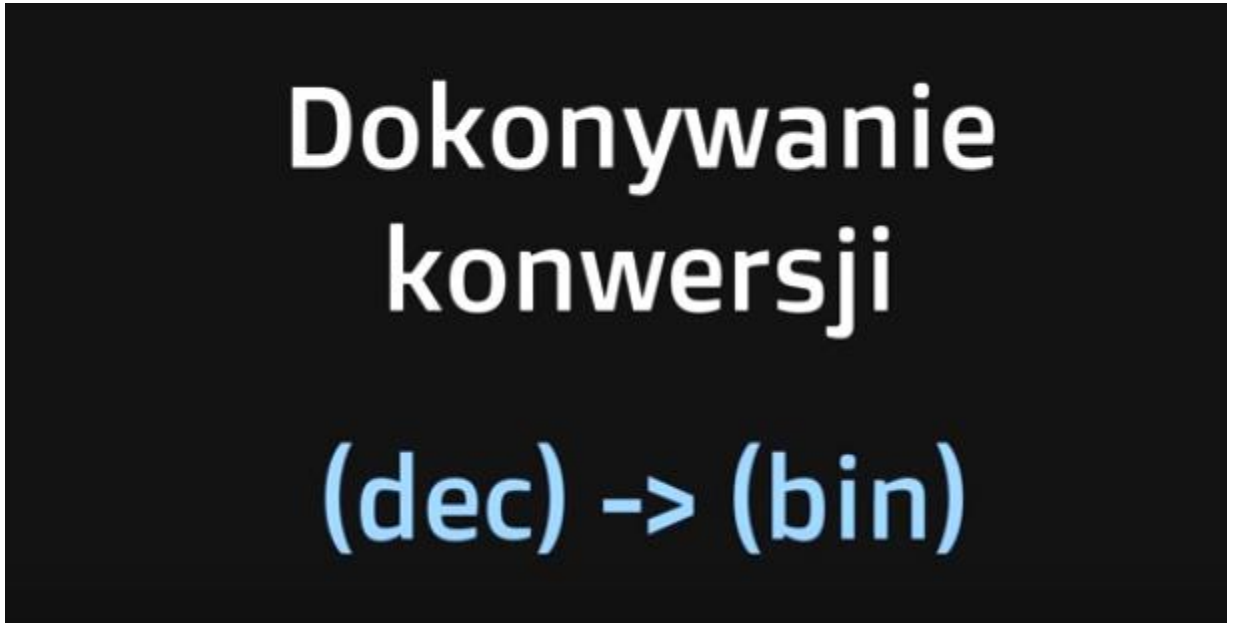


Reprezentacja liczb

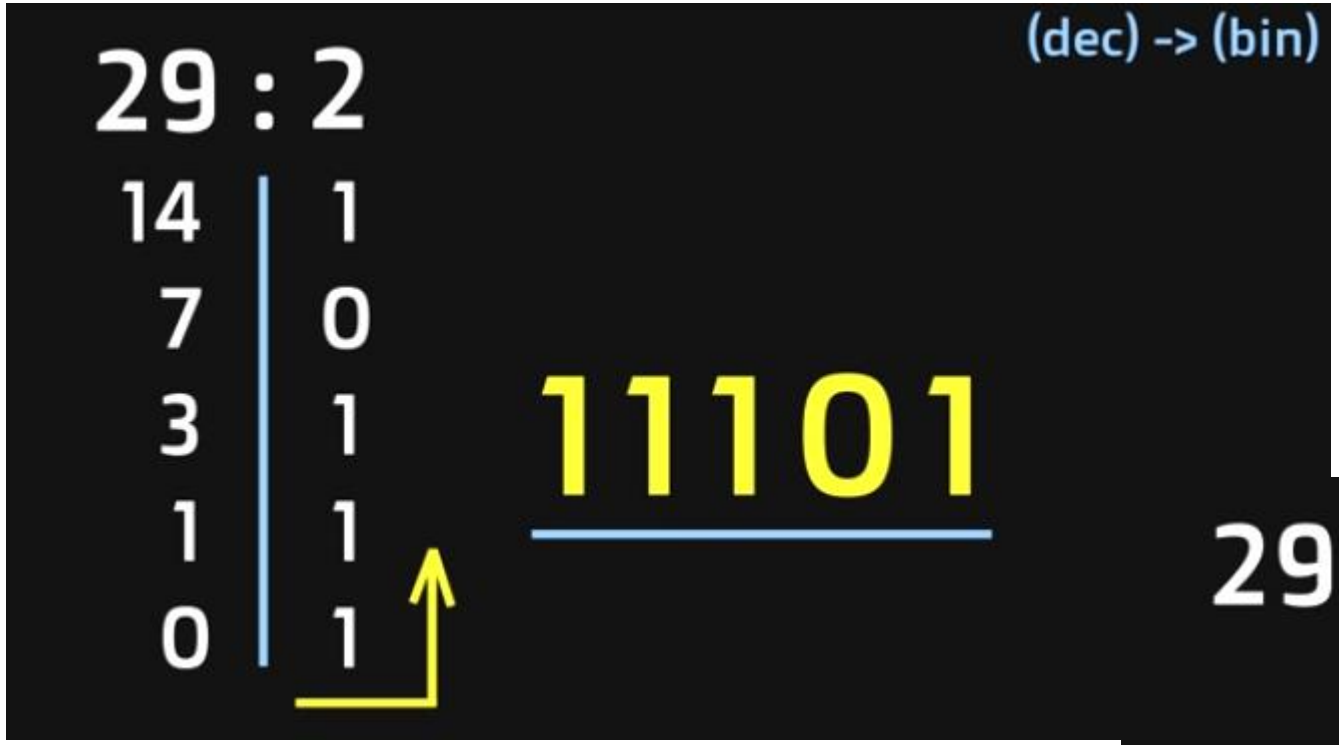


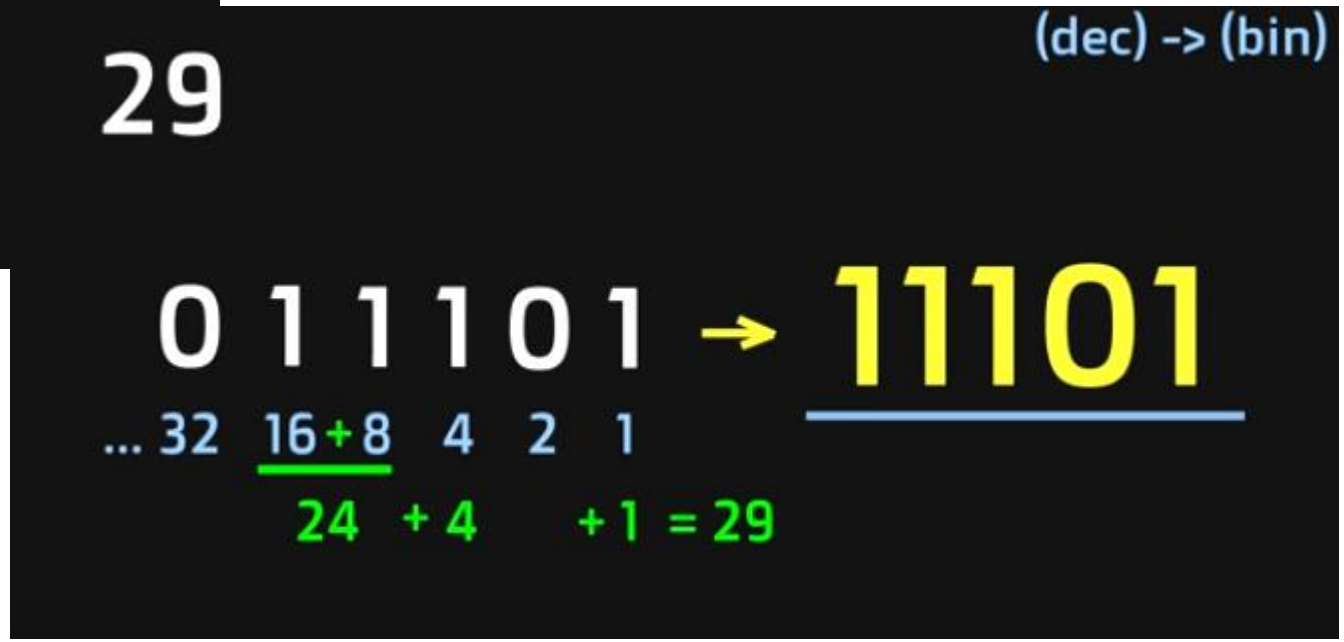


Reprezentacja liczb



Reprezentacja liczb





Reprezentacja liczb

System szesnastkowy
(heksadecymalny)

przydaje się do zapisu
naprawdę dużych liczb

Reprezentacja liczb

Adresy w pamięci RAM
001001010010110011001010

Adres MAC karty sieciowej:

dwójkowo:

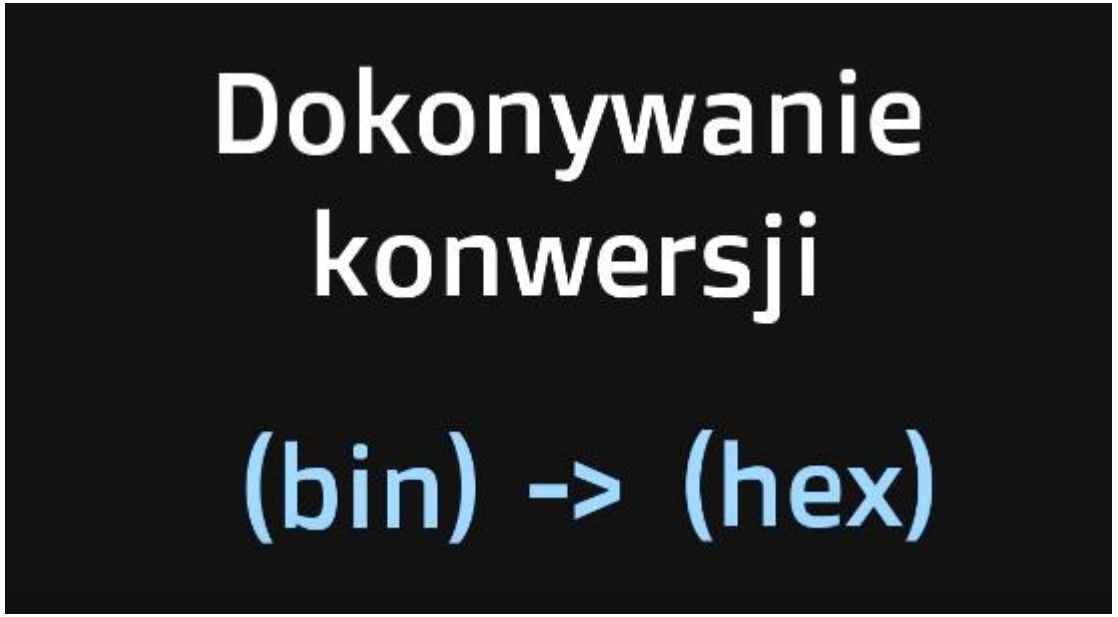
010011011110001011110100001001101101110101000011

szesnastkowo:

4DE2F426DD43

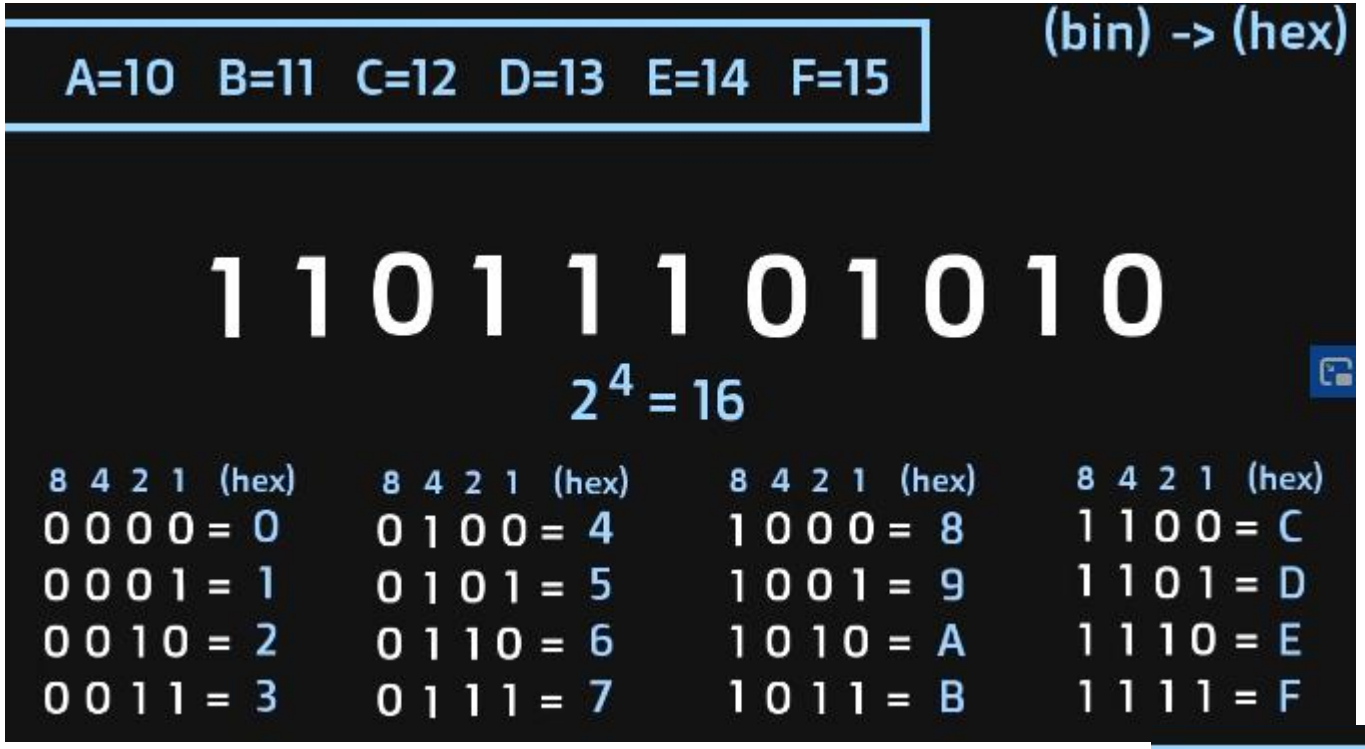
dziesiętnie:
85637154135363

Reprezentacja liczb

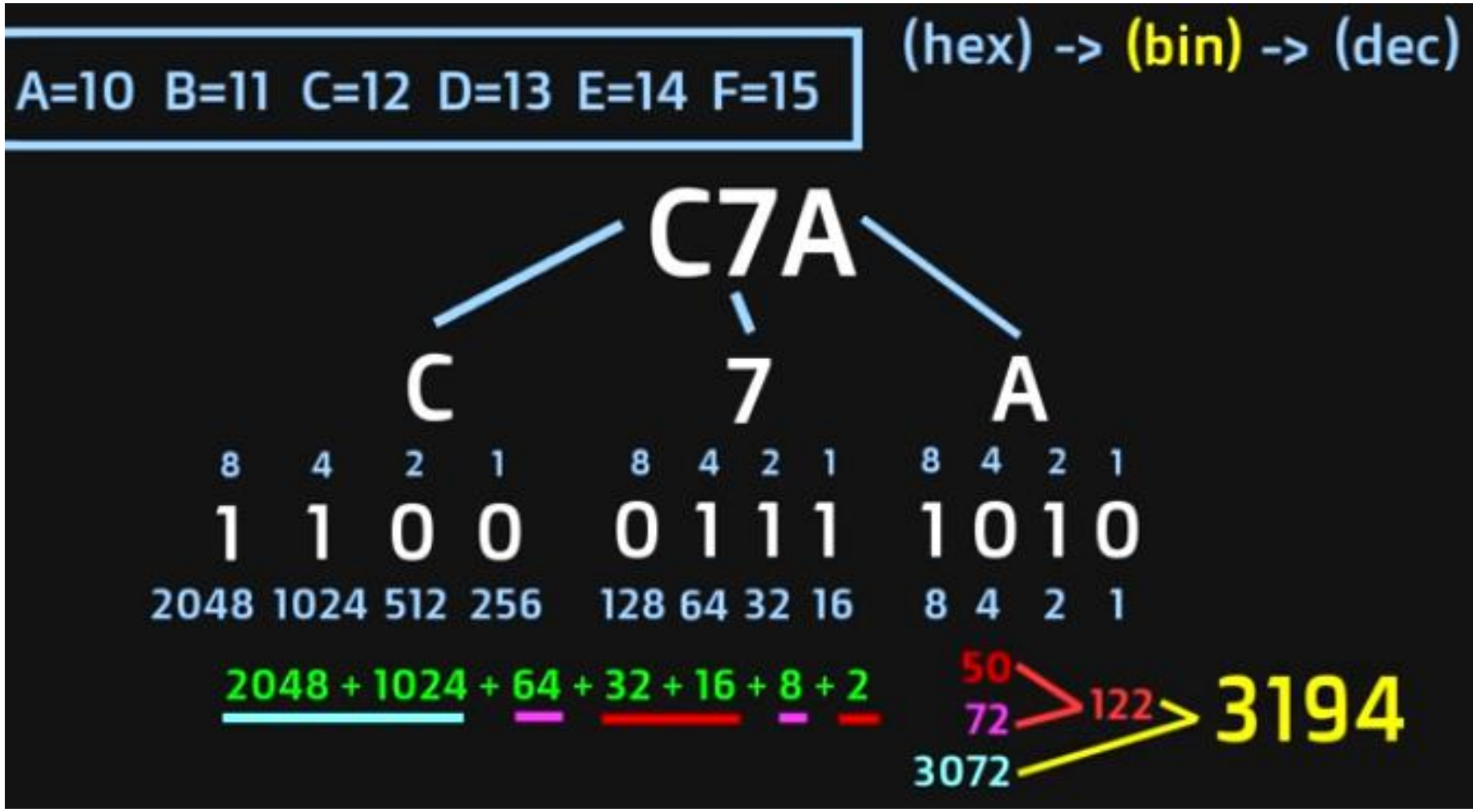


WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Reprezentacja liczb

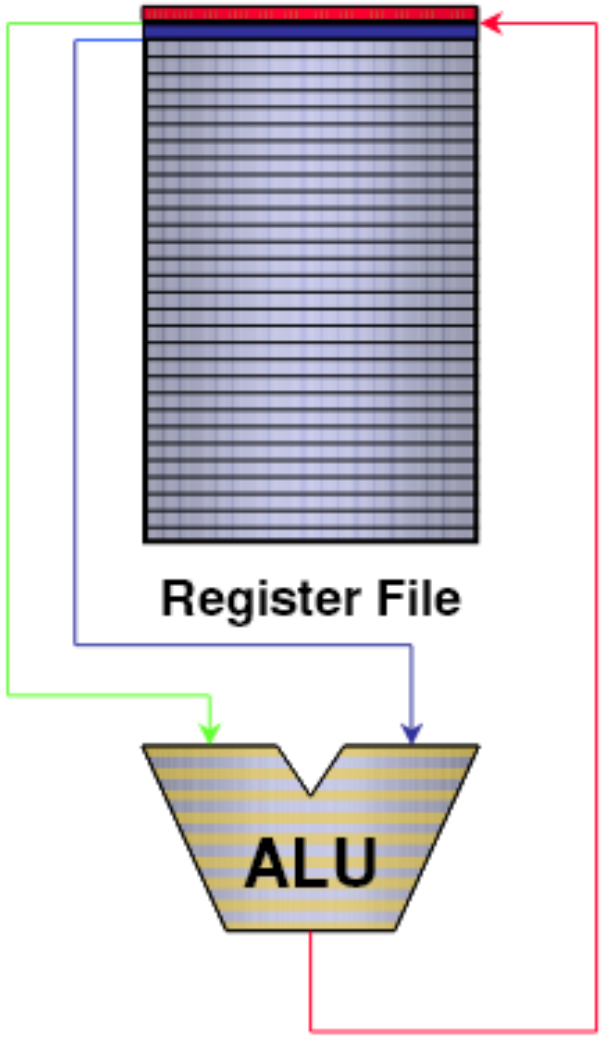


Reprezentacja liczb



WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Jednostka arytmetyczno-logiczna



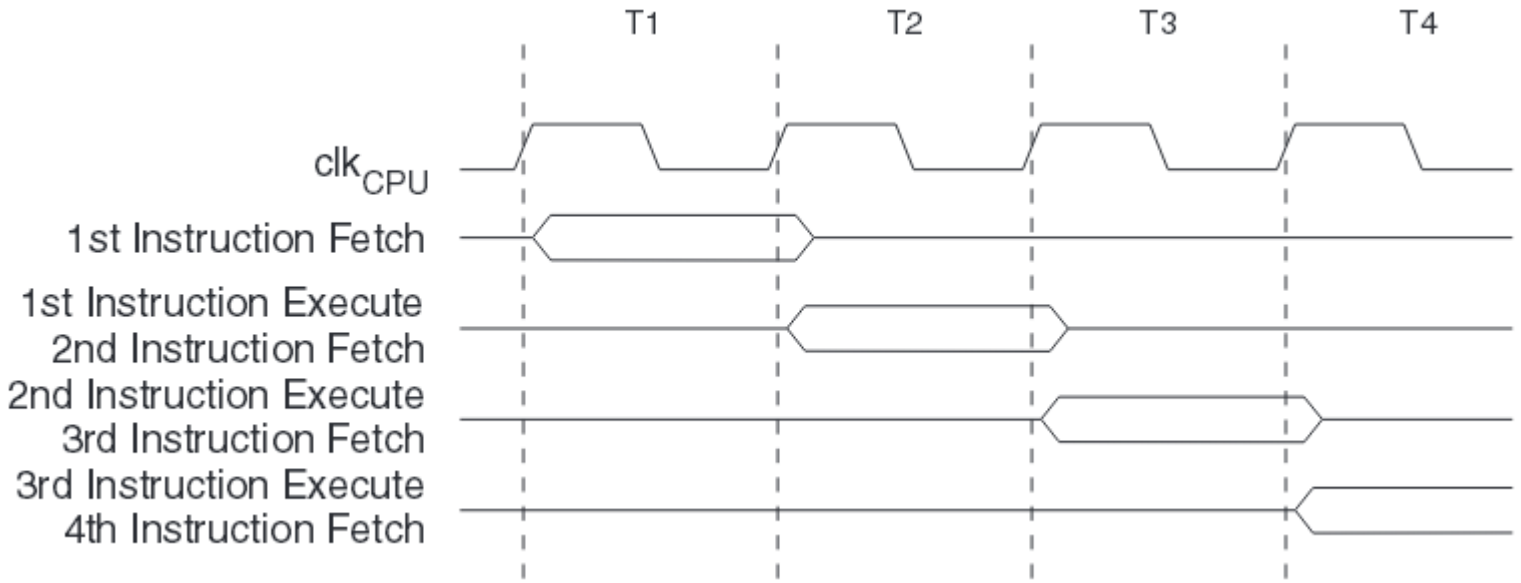
ALU- Arithmetic Logic Unit

Jest bezpośrednio połączona z 32 ośmiobitowymi rejestrami

Wykonuje operacje w jednym cyklu zegarowym:

- Arytmetyczne
- Logiczne
- Funkcje bitowe

Przetwarzanie potokowe (pipelining)

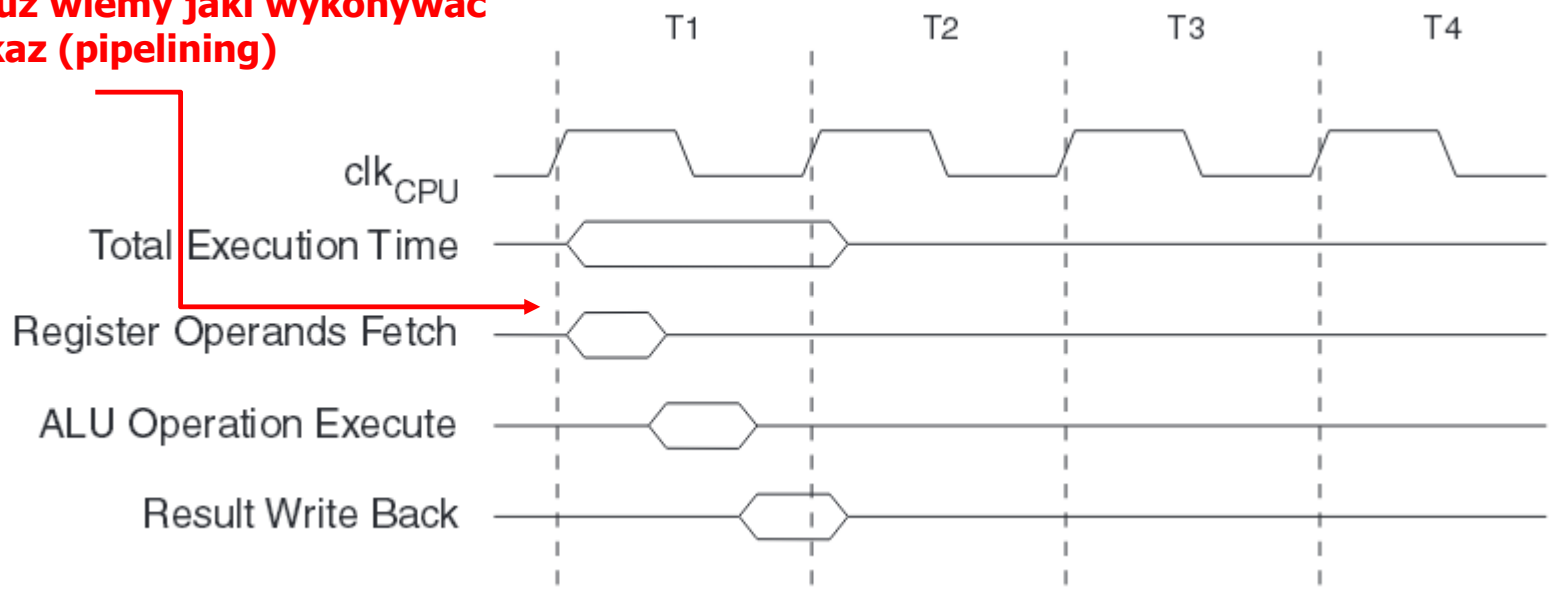


W AVR zastosowano przetwarzanie potokowe które polega na jednoczesnym wykonywaniu danej instrukcji i pobieraniu instrukcji następnej.
Ma to miejsce w jednym cyklu zegarowym.
AVR są układami 8 bitowymi jednak ich słowo rozkazowe ma 16 bitów.

Dla zegara 1MHz można osiągnąć 1 MIPS

Pojedynczy cykl pracy ALU

Tu już wiemy jaki wykonywać rozkaz (pipelining)



W pojedynczym cyklu zegara ALU wykonujemy operację na 2 rejestrach (źródło, przeznaczenie) i wynik przesyłamy do rejestru przeznaczenia

Zasady

Dodawanie binarne

Argument 1	Argument 2	Wynik	Przeniesienie
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Odejmowanie binarne

Argument 1	Argument 2	Wynik	Pożyczka
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Zasady

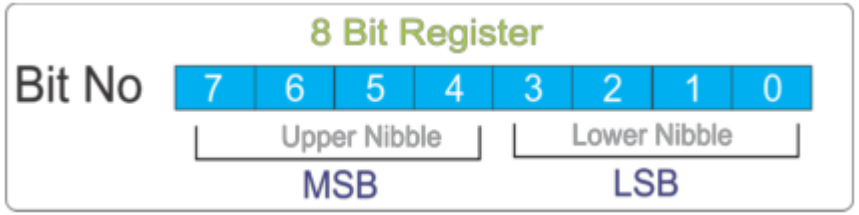
Operacje logiczne

Argument 1	Argument 2	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Dwa argumenty takie same

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania



Dodawanie dwóch liczb całkowitych bez znaku

5+8=13

```
ldi R16, 5     ; ładuj rejestr R16 wartością 5
ldi R17, 8     ; ładuj rejestr R16 wartością 8
add R16, R17
```

Wynik dodawania zapisywany jest w rejestrze R16

R16 = 13
00001101

	MSB				LSB			
	0	0	0	0	0	1	0	1
+	0	0	0	0	1	0	0	0
	0	0	0	0	1	1	0	1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	0	0	0	0	0	0

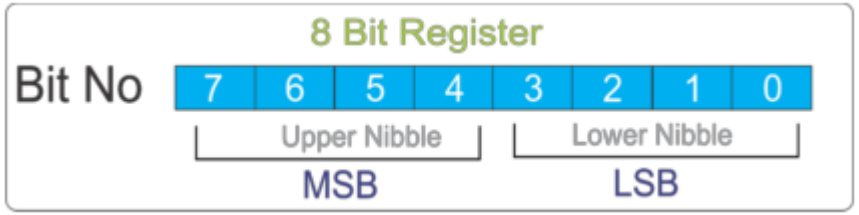
Operacja add modyfikuje flagi w rejestrze statusu

SREG

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania



Dodawanie dwóch liczb całkowitych bez znaku

15+35=50

```
ldi R16, 15     ; ładuj rejestr R16 wartością 15
ldi R17, 35     ; ładuj rejestr R16 wartością 35
add R16, R17
```

Wynik dodawania zapisywany jest w rejestrze R16

R16 = 50
00110010

MSB LSB

	0	0	0	0	1	1	1	1
+	0	0	1	0	0	0	1	1
	0	0	1	1	0	0	1	0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	1	0	0	0	0	0

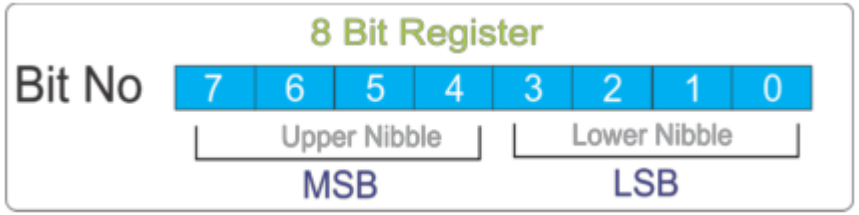
Operacja add modyfikuje flagi w rejestrze statusu

SREG

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania



Dodawanie dwóch liczb całkowitych bez znaku

100+28=128

```
ldi R16, 100    ; ładuj rejestr R16 wartością 100
ldi R17, 28     ; ładuj rejestr R16 wartością 28
add R16, R17
```

Wynik dodawania zapisywany jest w rejestrze R16

R16 = 128
10000000

	MSB				LSB			
	0	1	1	0	0	1	0	0
+	0	0	0	1	1	1	0	0
	(1)	0	0	0	0	0	0	0

N=1 otrzymana wartość w kodzie U2 byłaby ujemna,
V=1 otrzymana wartość w kodzie U2 byłaby poza zakresem (kod U2 przedział [-128; 127])

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	1	0	1	1	0	0

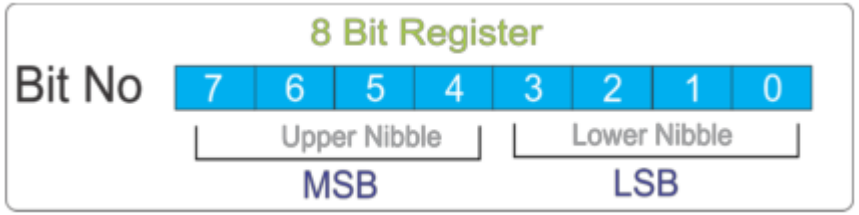
Operacja add modyfikuje flagi w rejestrze statusu

SREG

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania



Dodawanie dwóch liczb całkowitych bez znaku

147+124=271

```
ldi R16, 147    ; ładuj rejestr R16 wartością 147
ldi R17, 124    ; ładuj rejestr R16 wartością 124
add R16, R17
```

Wynik dodawania zapisywany jest w rejestrze R16

R16 = 15
00001111

Otrzymana wartość w rejestrze R16 jest poza zakresem nastąpiło przepełnienie sygnalizuje to C=1 w rejestrze SREG

	MSB				LSB			
	1	0	0	1	0	0	1	1
+	0	1	1	1	1	1	0	0
1	0	0	0	0	1	1	1	1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	0	0	0	0	0	1

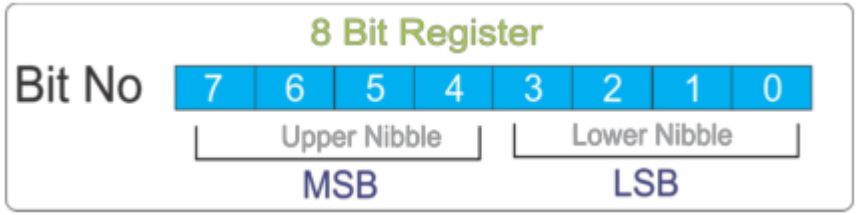
Operacja add modyfikuje flagi w rejestrze statusu

SREG

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania



Dodawanie dwóch liczb całkowitych bez znaku

255+255=510

```
ldi R16, 255    ; ładuj rejestr R16 wartością 255
ldi R17, 255    ; ładuj rejestr R16 wartością 255
add R16, R17
```

Wynik dodawania zapisywany jest w rejestrze R16

R16 = 254
11111110

	MSB				LSB			
	1	1	1	1	1	1	1	1
+	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0

N=1 otrzymana wartość w kodzie U2 byłaby ujemna,
V=1 otrzymana wartość w kodzie U2 byłaby poza zakresem (kod U2 przedział [-128; 127])

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	1	0	1	1	0	1

Operacja add modyfikuje flagi w rejestrze statusu

SREG

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania

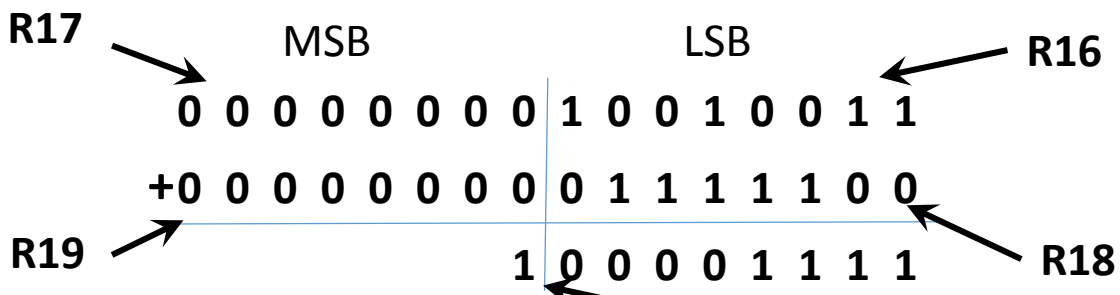
Jak wykorzystać znaczniki rejestru SREG
Przykładowo, rozkaz *adc* powoduje dodawanie zawartości dwóch rejestrów podobnie jak instrukcja *add*.
Jednak w przypadku *adc* wynik inkrementowany jest o 1, jeśli znacznik przeniesienia (C) jest ustawiony = 1. Stosując rozkaz *add* w połączeniu z *adc* możemy łatwo dodawać liczby 16- i więcej bitowe, np. parę rejestrów R19:R18 do pary R17:R16

147+124=271

```
ldi R16, low(147)    ; ładuj rejestr R16 młodszą cześć bitową wartości  147
ldi R17, high(147)   ; ładuj rejestr R17 starszą cześć bitową wartości  147
ldi R18, low(124)    ; ładuj rejestr R18 młodszą cześć bitową wartości  124
ldi R19, high(124)   ; ładuj rejestr R19 starszą cześć bitową wartości  124
add R16, R18         ; dodaj bajty młodsze
adc R17, R19         ; dodaj bajty starsze z przeniesieniem dodaje C z rejestru SREG
```

Wynik dodawania zapisywany jest w rejestrze R16:17

R16 = 15
00001111
R17 = 1
0000001
R16:R17
0000010000111
=271



Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	0	0	0	0	0	1

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

Przykłady dodawania

Dodawanie dwóch liczb całkowitych ze znakiem

W technice mikroprocesorowej najczęściej stosowany system kodowania liczb całkowitych ze znakiem wykorzystuje operację uzupełnienia do dwóch (kod U2; ang. two's complement). Wybór ten ma swoje uzasadnienie w łatwości przeprowadzania obliczeń na liczbach w tym formacie.

Operacje dodawania i odejmowania liczb w kodzie U2 przeprowadza się tak samo, jak na liczbach kodowanych naturalnie. Jest to podstawowa zaleta tego systemu i właśnie to kryterium decyduje o wyższości formatu U2 nad kodowaniem typu znak-moduł.

W przypadku operacji na liczbach ze znakiem znaczenia nabierają następujące bity rejestru SREG: V (informujący o przepełnieniu formatu U2), S (znak ujemny) i N (najstarszy bit ustawiony). Na podstawie ich stanu (z wykorzystaniem instrukcji br-) podejmowane mogą być decyzje o dalszym przebiegu programu.

Przykłady dodawania System U2

Zamiana liczb ujemnych

Zamieńmy liczbę −50 na system **U2**. Algorytm jest bardziej skomplikowany.
W pierwszym kroku wyznaczamy wartość bezwzględną z tej liczby:

$$|-50|=50$$

W drugim kroku otrzymaną liczbę zamieniamy na postać binarną:

$$50 = (110010)_2$$

W trzecim kroku przedstawiamy ją na ośmiu bitach:

$$50 = 00110010$$

W czwartym kroku negujemy wszystkie bity (każdy bit zamieniamy na przeciwny: zero na jedynkę, jedynkę na zero):

$$\sim(00110010) = 11001101$$

Na końcu zwiększamy otrzymaną postać o **1**:

$$11001101 + 1 = 11001110$$

W ten sposób otrzymaliśmy liczbę −50

zapisaną na **ośmiu bitach** w systemie **U2:**

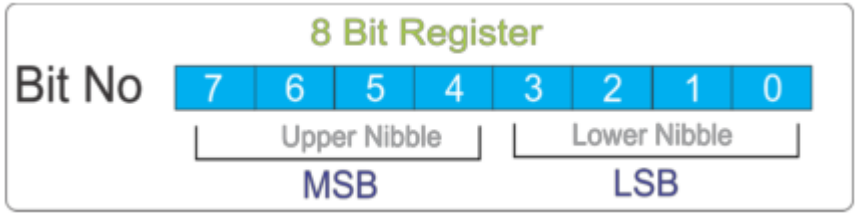
$$-50 = (11001110)_{U2}$$

Reprezentacja liczby binarnej w postaci dziesiętnej i kodzie U2

dec	bin	U2	dec	bin	U2	dec	bin	U2	dec	bin	U2	dec	bin	U2	dec	bin	U2	dec	bin	U2	dec	bin	U2
0	00000000	0	33	00100001	33	66	01000010	66	99	01100011	99	132	10000100	-124	165	10100101	-91	198	11000110	-58	231	11100111	-25
1	00000001	1	34	00100010	34	67	01000011	67	100	01100100	100	133	10000101	-123	166	10100110	-90	199	11000111	-57	232	11101000	-24
2	00000010	2	35	00100011	35	68	01000100	68	101	01100101	101	134	10000110	-122	167	10100111	-89	200	11001000	-56	233	11101001	-23
3	00000011	3	36	00100100	36	69	01000101	69	102	01100110	102	135	10000111	-121	168	10101000	-88	201	11001001	-55	234	11101010	-22
4	00000100	4	37	00100101	37	70	01000110	70	103	01100111	103	136	10001000	-120	169	10101001	-87	202	11001010	-54	235	11101011	-21
5	00000101	5	38	00100110	38	71	01000111	71	104	01101000	104	137	10001001	-119	170	10101010	-86	203	11001011	-53	236	11101100	-20
6	00000110	6	39	00100111	39	72	01001000	72	105	01101001	105	138	10001010	-118	171	10101011	-85	204	11001100	-52	237	11101101	-19
7	00000111	7	40	00101000	40	73	01001001	73	106	01101010	106	139	10001011	-117	172	10101100	-84	205	11001101	-51	238	11101110	-18
8	00001000	8	41	00101001	41	74	01001010	74	107	01101011	107	140	10001100	-116	173	10101101	-83	206	11001110	-50	239	11101111	-17
9	00001001	9	42	00101010	42	75	01001011	75	108	01101100	108	141	10001101	-115	174	10101110	-82	207	11001111	-49	240	11110000	-16
10	00001010	10	43	00101011	43	76	01001100	76	109	01101101	109	142	10001110	-114	175	10101111	-81	208	11010000	-48	241	11110001	-15
11	00001011	11	44	00101100	44	77	01001101	77	110	01101110	110	143	10001111	-113	176	10110000	-80	209	11010001	-47	242	11110010	-14
12	00001100	12	45	00101101	45	78	01001110	78	111	01101111	111	144	10010000	-112	177	10110001	-79	210	11010010	-46	243	11110011	-13
13	00001101	13	46	00101110	46	79	01001111	79	112	01110000	112	145	10010001	-111	178	10110010	-78	211	11010011	-45	244	11110100	-12
14	00001110	14	47	00101111	47	80	01010000	80	113	01110001	113	146	10010010	-110	179	10110011	-77	212	11010100	-44	245	11110101	-11
15	00001111	15	48	00110000	48	81	01010001	81	114	01110010	114	147	10010011	-109	180	10110100	-76	213	11010101	-43	246	11110110	-10
16	00010000	16	49	00110001	49	82	01010010	82	115	01110011	115	148	10010100	-108	181	10110101	-75	214	11010110	-42	247	11110111	-9
17	00010001	17	50	00110010	50	83	01010011	83	116	01110100	116	149	10010101	-107	182	10110110	-74	215	11010111	-41	248	11111000	-8
18	00010010	18	51	00110011	51	84	01010100	84	117	01110101	117	150	10010110	-106	183	10110111	-73	216	11011000	-40	249	11111001	-7
19	00010011	19	52	00110100	52	85	01010101	85	118	01110110	118	151	10010111	-105	184	10111000	-72	217	11011001	-39	250	11111010	-6
20	00010100	20	53	00110101	53	86	01010110	86	119	01110111	119	152	10011000	-104	185	10111001	-71	218	11011010	-38	251	11111011	-5
21	00010101	21	54	00110110	54	87	01010111	87	120	01111000	120	153	10011001	-103	186	10111010	-70	219	11011011	-37	252	11111100	-4
22	00010110	22	55	00110111	55	88	01011000	88	121	01111001	121	154	10011010	-102	187	10111011	-69	220	11011100	-36	253	11111101	-3
23	00010111	23	56	00111000	56	89	01011001	89	122	01111010	122	155	10011011	-101	188	10111100	-68	221	11011101	-35	254	11111110	-2
24	00011000	24	57	00111001	57	90	01011010	90	123	01111011	123	156	10011100	-100	189	10111101	-67	222	11011110	-34	255	11111111	-1
25	00011001	25	58	00111010	58	91	01011011	91	124	01111100	124	157	10011101	-99	190	10111110	-66	223	11011111	-33			
26	00011010	26	59	00111011	59	92	01011100	92	125	01111101	125	158	10011110	-98	191	10111111	-65	224	11100000	-32			
27	00011011	27	60	00111100	60	93	01011101	93	126	01111110	126	159	10011111	-97	192	11000000	-64	225	11100001	-31			
28	00011100	28	61	00111101	61	94	01011110	94	127	01111111	127	160	10100000	-96	193	11000001	-63	226	11100010	-30			
29	00011101	29	62	00111110	62	95	01011111	95	128	10000000	-128	161	10100001	-95	194	11000010	-62	227	11100011	-29			
30	00011110	30	63	00111111	63	96	01100000	96	129	10000001	-127	162	10100010	-94	195	11000011	-61	228	11100100	-28			
31	00011111	31	64	01000000	64	97	01100001	97	130	10000010	-126	163	10100011	-93	196	11000100	-60	229	11100101	-27			
32	00100000	32	65	01000001	65	98	01100010	98	131	10000011	-125	164	10100100	-92	197	11000101	-59	230	11100110	-26			

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania



Dodawanie dwóch liczb całkowitych ze znakiem

100+(-28)=72

```
ldi R16, 100     ; ładuj rejestr R16 wartością 100
ldi R17, -28     ; ładuj rejestr R16 wartością -28, (przekształcenie stałej bezpośredniej -20 na kod U2 przeprowadzane jest
add R16, R17                automatycznie przez translator!)
```

Wynik dodawania zapisywany jest w rejestrze R16

R16 = 72
01001000

	MSB				LSB				
	0	1	1	0	0	1	0	0	100
+	1	1	1	0	0	1	0	0	228
(1)	0	1	0	0	1	0	0	0	

-28 zapisana w kodzie U2 daje wartość dziesiętną 228,

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	0	0	0	0	0	1

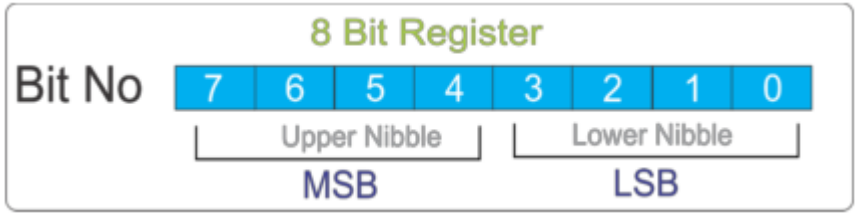
Operacja add modyfikuje flagi w rejestrze statusu

SREG

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania



Dodawanie dwóch liczb całkowitych ze znakiem

10+(-20)= -10

```
ldi R16, 10     ; ładuj rejestr R16 wartością 10
ldi R17, -20    ; ładuj rejestr R16 wartością -10, (przekształcenie stałej bezpośredniej -10 na kod U2 przeprowadzane jest
add R16, R17                automatycznie przez translator!)
```

Wynik dodawania zapisywany jest w rejestrze R16

R16 = 246
11110110

$(11110110)_{U2} = -10$

	MSB				LSB				
	0	0	0	0	1	0	1	0	10
+	1	1	1	0	1	1	0	0	236
	1	1	1	1	0	1	1	0	

-28 zapisana w kodzie U2 daje wartość dziesiętną 228,

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	1	1	0	1	0	0

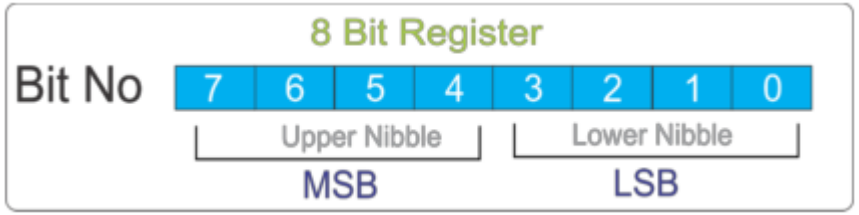
Operacja add modyfikuje flagi w rejestrze statusu

SREG

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania



Dodawanie dwóch liczb całkowitych ze znakiem

(-10)+(-120)= -130

```
ldi R16, -10     ; ładuj rejestr R16 wartością 10
ldi R17, -120    ; ładuj rejestr R16 wartością -120, (przekształcenie stałej bezpośredniej -120 na kod U2 przeprowadzane jest
add R16, R17                automatycznie przez translator!)
```

Wynik dodawania zapisywany jest w rejestrze R16

	MSB				LSB				
	1	1	1	1	0	1	1	0	245
+	1	0	0	0	1	0	0	0	136
1	0	1	1	1	1	1	1	0	

R16 = 126
01111110

$(01111110)_{U2} = 126$

-28 zapisana w kodzie U2 daje wartość dziesiętną 228,

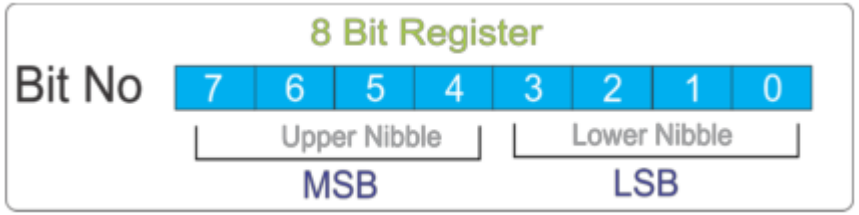
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	0	1	1	0	0	1

Operacja add modyfikuje flagi w rejestrze statusu

SREG

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

Przykłady dodawania



Problem!! **(-10)+(-120)= -130**

```
ldi R16, -10      ; ładuj rejestr R16 wartością -10
ldi R17, -120     ; ładuj rejestr R17 wartością -120, (przekształcenie stałej bezpośredniej -120 na kod U2 przeprowadzane jest
add R16, R17                automatycznie przez translator!)
```

Wynik dodawania zapisywany jest w rejestrze R16

R16 = 126
01111110

$(01111110)_{U2} = 126$

Oczekujemy wyniku –130 , ale pamiętajmy, że na pojedynczym bajcie w kodowaniu U2 zapisać można co najwyżej liczbę –128. W wyniku działania powyższych rozkazów w rejestrze R16 znalazła się wartość 126, a w rejestrze SREG ustawiono znaczniki V, S i C. Skoro umówiliśmy się, że pracujemy na liczbach w kodowaniu U2, wartość znacznika V jest istotna. Wiemy zatem, że wynik jest niepoprawny (bo V=1), ale możemy być jednocześnie pewni, że właściwy wynik powinien mieć wartość ujemną (bo S=1). Jeśli taka informacja jest zadowalająca, możemy na tym zakończyć.

Przykłady odejmowania

Rozkazów odejmujących (sub, sbc, subi i sbci) nie będziemy już tak szczegółowo omawiać – od instrukcji dodawania odróżnia je automatyczna zamiana dodajnika na jego uzupełnienie dwójkowe (co w efekcie każe traktować go jako odjemnik) i odmienny sposób ustawiania znaczników w rejestrze statusu. Przede wszystkim należy pamiętać, że w przypadku rozkazów odejmujących bit C z rejestru statusu powinien być interpretowany jako znacznik pożyczki do najstarszej pozycji wyniku.

Odejmowanie liczb mniejszych niż –128 powoduje, że translator wyświetla ostrzeżenia o przekroczeniu zakresu instrukcji odejmującej. Ma on oczywiście rację – liczb mniejszych od –128 nie można zapisać w kodzie U2 na jednym bajcie, ale ostrzeżenie to możemy w tym przypadku zignorować.

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady dodawania i odejmowania

Dodawanie dwóch liczb całkowitych ze znakiem

10+(-20)= -10

```
ldi R16, 10      ; ładuj rejestr R16 wartością 10
ldi R17, -20     ; ładuj rejestr R16 wartością -10
add R16, R17     ; dodaj do rejestru R16 rejestr R17
```

R16 = 246
11110110

	MSB				LSB				
	0	0	0	0	1	0	1	0	10
+	1	1	1	0	1	1	0	0	236
	1	1	1	1	0	1	1	0	

$(11110110)_{U2} = -10$

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	1	1	0	1	0	0

Operacja add modyfikuje flagi w rejestrze statusu SREG

Odejmowanie dwóch liczb całkowitych

10-20= -10

```
ldi R16, 10      ; ładuj rejestr R16 wartością 10
ldi R17, -20     ; ładuj rejestr R16 wartością 20,
sub R16, R17     ; odejmij  do rejestru R16 rejestr R17
```

R16 = 246
11110110

pożyczka

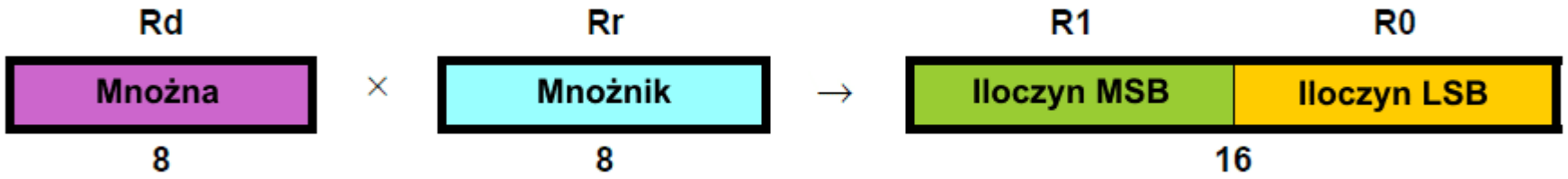
		MSB				LSB				
	1	0	0	0	0	1	0	1	0	10
-		0	0	0	1	0	1	0	0	20
		1	1	1	1	0	1	1	0	

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	0	1	0	1	0	1

Operacja add modyfikuje flagi w rejestrze statusu SREG

Przykłady mnożenia

Format kodowania – czyli sposób interpretacji przetwarzanej liczby – znany jest wyłącznie programiście. Choć przy operacjach dodawania i odejmowania rodzaj kodowania determinuje jedynie interpretację znaczników w rejestrze statusu (o ile stosujemy oczywiście kodowanie naturalne lub U2), to w przypadku mnożenia operacje muszą być przeprowadzane odmiennie dla liczb ze znakiem i bez niego. Aby problem zminimalizować, listę rozkazów AVR wyposażono w instrukcje mnożące zarówno liczby w formacie bez znaku (w naturalnym kodzie binarnym – rozkaz **mul**), ze znakiem (w kodzie U2 – **muls**), jak i mieszane (pierwsza w kodzie U2, druga w naturalnym kodzie binarnym – **mulsu**).



mul - Mnożna Rd i mnożnik Rr są dwoma rejestrami zawierającymi liczby bez znaku. 16-bitowy iloczyn bez znaku jest umieszczany w R1 (górny bajt, MSB) i R0 (dolny bajt, LSB). Zwróć uwagę, że jeśli mnożna lub mnożnik zostaną wybrane z R0 lub R1, to wynik mnożenia nadpisze zawartość tych rejestrów.

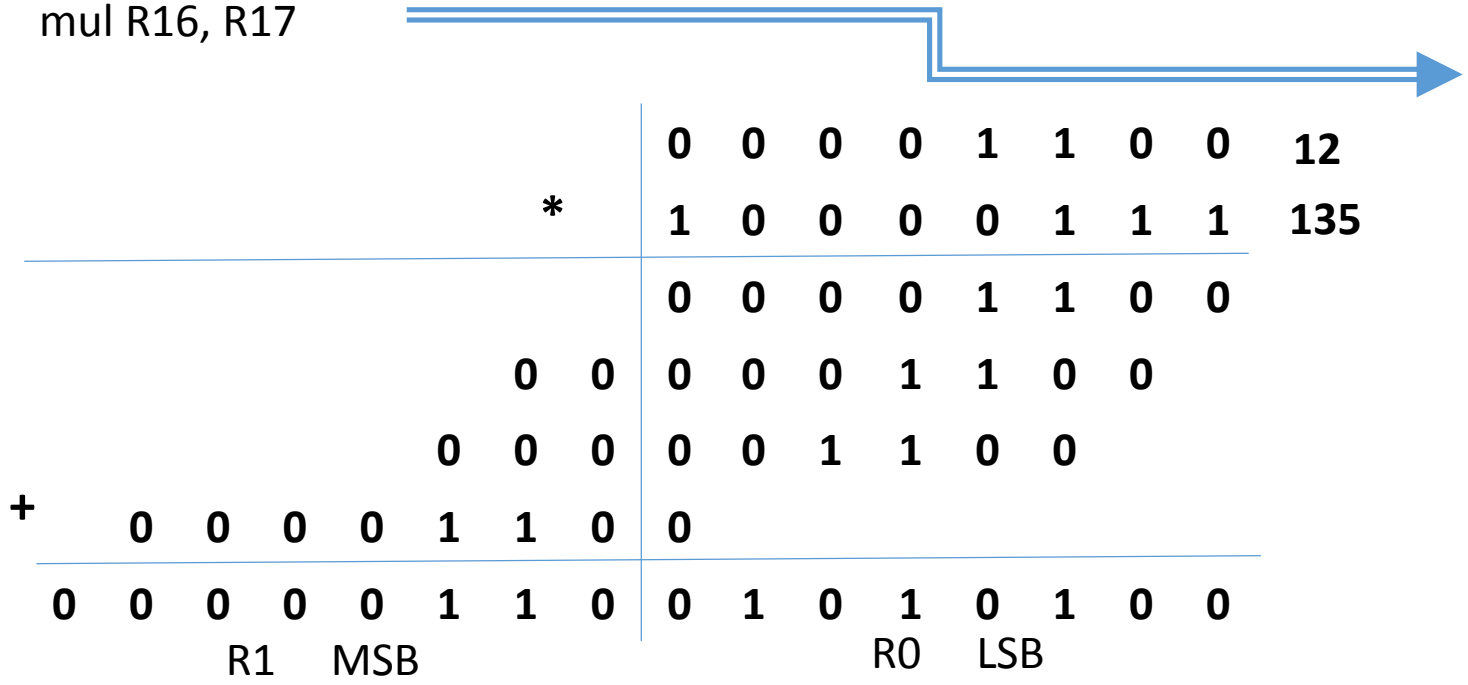
WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Przykłady mnożenia

Mnożenie dwóch liczb całkowitych bez znaku

12*135= 1620

```
ldi R16, 12      ; ładuj rejestr R16 wartością 12
ldi R17, 135     ; ładuj rejestr R17 wartością 135
mul R16, R17
```



Wynik mnożenia zapisywany jest w rejestrze R0 i R1

R0 = 84
01010100
R1 = 6
000001100

Z: jest ustawiany jeśli wynik jest zerowy R1:R2=0x0000
C: jest ustawiony jeśli bit Msb iloczynu (R1(7)) jest 1 w przeciwnym wypadku jest zerowany

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I	T	H	S	V	N	Z	C
0	0	0	0	0	0	0	0

Bit 7 – I: zezwolenie na przerwania (sei, cli)
Bit 6 – T: zachowywanie/odtwarzanie kopii bitu (bst, bld)
Bit 5 – H: przeniesie pomiędzy LSB a MSB
Bit 4 – S: bit znaku
Bit 3 – V: wskaźnik przepełnienia uzupełnienia do dwóch
Bit 2 – N: wskaźnik wartości ujemnej
Bit 1 – Z: wskaźnik zera
Bit 0 – C: wskaźnik przeniesienia

Operacja mul modyfikuje tylko flagi Z i C w rejestrze statusu SREG