POLITECHNIKA GDAŃSKA

Wydział Elektroniki, Telekomunikacji i Informatyki
Studia podyplomowe
„Aplikacje i usługi internetowe”

Podstawy programowania

Cz. I

Opracował dr inż. Andrzej Jędruch

Listopad 2017

TYPOWE KOMPUTERY I PROCESORY OD R.1946

Rok	Nazwa	Opis
1946	ENIAC	Pierwszy komputer elektroniczny
1951	UNIVAC	Pierwszy komputer przeznaczony do sprzedaży
1964	IBM 360	Typowy komputer o dużej wydajności (ang. mainframe)
1965	PDP-8	Pierwszy minikomputer
1971	Intel 4004	Pierwszy mikroprocesor
1977	Apple II	Pierwszy komputer osobisty
1981	IBM PC (procesor Intel 8088 i system MS-DOS)	Konstrukcja procesora 8086/88 rozwijana do chwili obecnej
2003	Intel Pentium 4	„Ostatni” procesor jednordzeniowy
2011	Intel Core i7	Typowy procesor wielordzeniowy

SYSTEMY WBUDOWANE – KOMPUTERY, KTÓRYCH NIE WIDAĆ

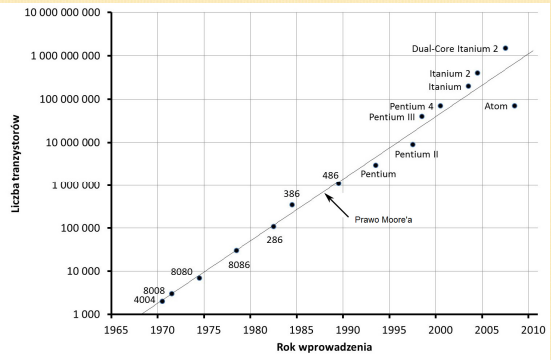
- Występują w wielu urządzeniach, ale użytkownik nie jest świadomy ich istnienia, np.
- w telefonach komórkowych,
- w samochodach, w lodówkach, w zabawkach,
- w aparaturze medycznej,
- i w wielu innych urządzeniach.



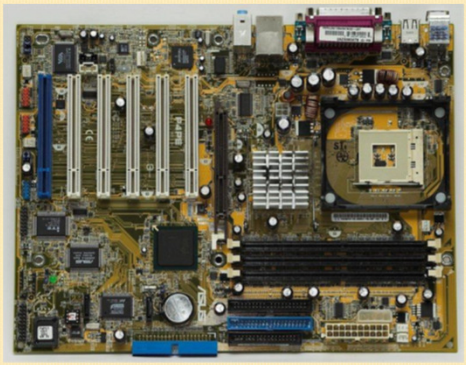
„PRAWO” MOORE’A (1)

- Przyjmuje się, że moc obliczeniowa komputerów podwaja się co 24 miesiące; podobnie liczba tranzystorów . . .

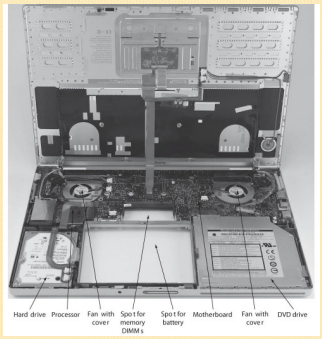
„PRAWO” MOORE’A (2)



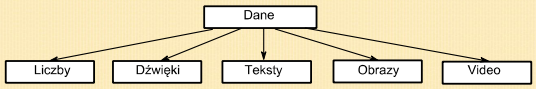
PŁYTA GŁÓWNA KOMPUTERA



WNĘTRZE LAPTOPA

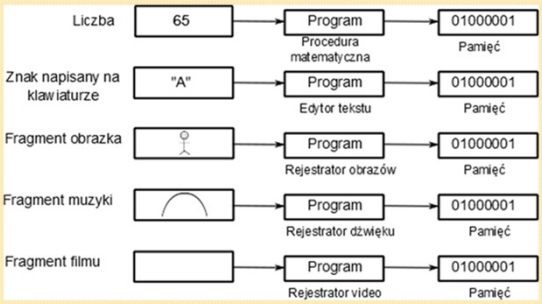


TYPY DANYCH PRZETWARZANYCH W KOMPUTERZE



- Przed zapisaniem w komputerze wszystkie typy danych są transformowane do jednolitej reprezentacji. W trakcie odczytywania przywracana jest pierwotna postać danych.

REPREZENTACJA DANYCH W PAMIĘCI KOMPUTERA



BITY, BAJTY, SŁOWA … (1)

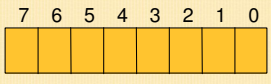
- Najmniejszą jednostką danych, która może być przechowywana w komputerze jest bit (cyfra w systemie dwójkowym).
- Specyficzne cechy elementów stosowanych w urządzeniach cyfrowych spowodowały, że przechowywane i przetwarzane informacje mają postać ciągów złożonych z zer i jedynek.
- Tak więc elementarną jednostką przetwarzania informacji jest cyfra systemu dwójkowego czyli bit — bit może przyjmować jedną z dwóch wartości: 0 albo 1.

BITY, BAJTY, SŁOWA … (2)

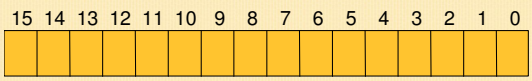
- Bit jest najmniejszą możliwą jednostką informacji i w pewnych zastosowaniach może wystarczająco dokładnie odwzorowywać sytuację, np. drzwi otwarte albo zamknięte. W przypadku liczb pojedynczy bit pozwala tylko na zapisanie dwóch liczb: 0 albo 1, co jest całkowicie niepraktyczne.
- Z tego powodu tworzy się zespoły bitów:
  - 8 bitów — bajt (ang. byte)
  - 16 bitów słowo (ang. word)
  - 32 bity — podwójne słowo (ang. double word),
  - 64 bity — poczwórne słowo (ang. quad word), itd.

BITY, BAJTY, SŁOWA … (3)

- Niekiedy terminem „słowo” określa się ciągi o długości 32 bitów.



bajt



słowo (ang. word)

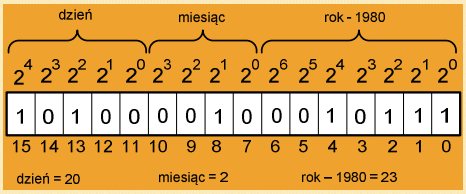
BITY, BAJTY, SŁOWA … (4)

- Bajty lub ich zespoły nie zawierają żadnej, skojarzonej z nimi, informacji o ich znaczeniu.
- Ciąg złożony z 16 bitów (2 bajty) może reprezentować liczbę naturalną, może opisywać datę, może być instrukcją opisującą czynności wykonywane przez układy elektroniczne i może być interpretowany na wiele innych sposobów.
- W praktyce programowania stosowane są często ciągi 32-bitowe (4 bajty), które m.in. mogą być interpretowane jako liczby z przedziału

<–2 147 483 648, + 2 147 483 647>

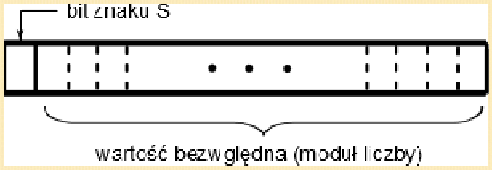
BITY, BAJTY, SŁOWA … (5)

- Przykład: ciąg 16 bitów 1010000100010111 może być traktowany jako liczba o wartości dziesiętnej 41239, ale może również stanowić datę kodowaną wg poniższego schematu.



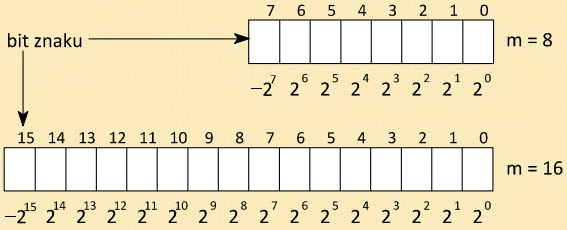
LICZBY ZE ZNAKIEM I BEZ ZNAKU (1)

- W celu określenia znaku liczby przechowywanej w pamięci przyjęto, że jeśli skrajny lewy bit jest równy 1, to liczba jest ujemna, w przeciwnym razie jest dodatnia. Podany niżej schemat ilustruje jeden z dwóch sposobów reprezentowania liczb znany jako „znak–moduł”.



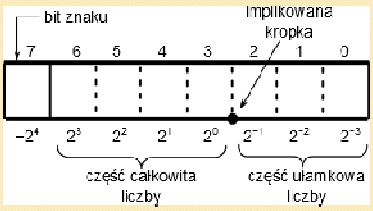
LICZBY ZE ZNAKIEM I BEZ ZNAKU (2)

- W komputerach stosowany jest jednak powszechnie inny sposób kodowania liczb ze znakiem oznaczony symbolem U2.



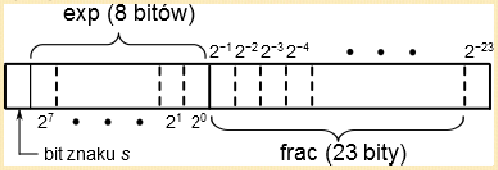
LICZBY ZE ZNAKIEM I BEZ ZNAKU (3)

- Procesor może też wykonywać działania na liczbach mieszanych zawierających część całkowitą i ułamkową – programowanie obliczeń dla liczb zapisanych w takich formatach jest dość kłopotliwe.



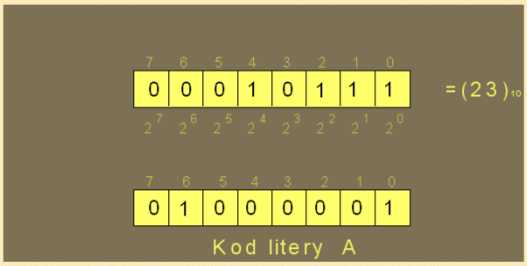
LICZBY ZE ZNAKIEM I BEZ ZNAKU (4)

- Procesor może też wykonywać działania na liczbach w formatach zmiennoprzecinkowych, które zawierają właściwą liczbę znormalizowaną do przedziału <1, 2) lub (–2, –1> oraz drugą liczbę, który jest wykładnikiem potęgi 2. Iloczyn liczby znormalizowanej i potęgi określa wartość liczby. Liczba 0 traktowana jest jako wartość specjalna.



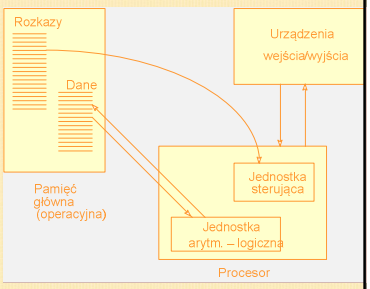
BITY, BAJTY, SŁOWA … (6)

- Ciąg 8-bitowy reprezentuje często znak w kodzie ASCII, ale może również oznaczać liczbę:



MODEL PROGRAMOWY KOMPUTERA (1)

- Zasadniczą część komputera stanowi procesor, który ściśle współpracuje z pamięcią główną (nazywanej także operacyjną). W pamięci przechowywane są dane i wyniki, a także instrukcje (rozkazy) dla procesora.



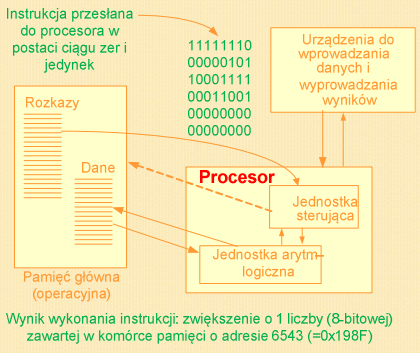
MODEL PROGRAMOWY KOMPUTERA (2)

- Procesor pobiera instrukcje z pamięci i wykonuje odpowiednie działania na danych. Ciąg instrukcji, który opisuje sposób wykonania pewnego zadania określamy nazwą *programu*.
- Współczesne procesory są urządzeniami bardzo skomplikowanymi (liczba tranzystorów w typowych procesorach dochodzi do miliarda).
- Dla potrzeb programowania można posługiwać się modelem komputera, który sformułował matematyk amerykański von Neumann (r. 1945). Model ten jest nadal podstawą konstrukcji prawie wszystkich komputerów, chociaż ich funkcje wewnętrzne są daleko bardziej skomplikowane.

MODEL PROGRAMOWY KOMPUTERA (3)

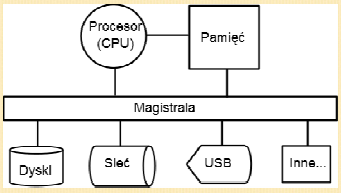
- Układy elektroniczne komputera (procesor) pozwalają rozpoznać i wykonać bardzo ograniczony zbiór prostych instrukcji — każdy program napisany w języku wysokiego poziomu musi zostać przekształcony na takie instrukcje, aby mógł zostać wykonany.
- Rozmiar ww. zbioru instrukcji wynika z dążenia do ograniczenia kosztu i złożoności układów elektronicznych.

MODEL PROGRAMOWY KOMPUTERA (4)



MODEL PROGRAMOWY KOMPUTERA (5)

- Model programowy komputera nie określa budowy wewnętrznej komputera, która zazwyczaj jest bardzo skomplikowana. Istotną rolę w przesyłaniu sygnałów między podzespołami komputera odgrywają drogi połączeniowe, nazywane magistralami lub szynami (ang. bus).



PROCESORY WIELORDZENIOWE I WIELOWĄTKOWE (1)

- Postęp w dziedzinie konstrukcji układów elektronicznych umożliwił w ciągu ostatnich 10 lat wytwarzanie zespołów 2, 4 i więcej procesorów umieszczonych w jednej obudowie. Tego rodzaju zespoły nazywane są procesorami dwu-, cztero-, ośmiordzeniowymi, itd.
- Zatem procesor czterordzeniowy to po prostu cztery samodzielne procesory zamknięte w jednej obudowie. Każdy z tych procesorów ma zdolność wykonywania oddzielnego programu.

PROCESORY WIELORDZENIOWE I WIELOWĄTKOWE (2)

- Bardziej skomplikowana jest koncepcja procesorów wielowątkowych. Procesor dwuwątkowy stanowi pojedynczy procesor, który poprzez rozbudowanie niektórych układów wewnętrznych ma zdolność jednoczesnego wykonywania dwóch oddzielnych programów.
- Wydajność procesora dwuwątkowego jest mniejsza niż procesora dwurdzeniowego.
- Przykładowo, procesor Intel Core i7 ma 4 rdzenie, a każdy rdzeń może pracować w trybie dwuwątkowym. W rezultacie procesor może wykonywać jednocześnie 8 programów.

PRZEDROSTKI DZIESIĘTNE I BINARNE (1)

- Zazwyczaj przy określaniu rozmiarów pamięci stosowane są przedrostki będące potęgami 2, a nie 10 jak w układzie SI. Dla przedrostków binarnych wprowadzone zmodyfikowane oznaczenia, które są jednak są rzadko używane (zob. tabela w następnym slajdzie).
- Sytuacja ta może być źródłem nieporozumień, np. producenci dysków używają przedrostków z układu SI, co oznacza, że dysk o pojemności 2 TB zawiera 2 000 000 000 000 bajtów. Pojemność tego samego dysku wyrażona w tebibajtach wynosi 1,82 TiB. Jednak system operacyjny podaje zwykle wartość 1,82 TB.

PRZEDROSTKI DZIESIĘTNE I BINARNE (2)

Przedrostki dziesiętne		
kilobajt	kB	$10^3 = 1000$
megabajt	MB	$10^6 = 1\,000\,000$
gigabajt	GB	$10^9 = 1\,000\,000\,000$
terabajt	TB	$10^{12} = 1\,000\,000\,000\,000$

Przedrostki binarne		
kibibajt	KiB	$2^{10} = 1024$
mebibajt	MiB	$2^{20} = 1\,048\,576$
gibibajt	GiB	$2^{30} = 1\,073\,741\,824$
tebibajt	TiB	$2^{40} = 1\,099\,511\,627\,776$

PAMIĘĆ GŁÓWNA (OPERACYJNA) (1)

- Pamięć główna (operacyjna) komputera składa z dużej liczby komórek (np. kilka miliardów), a każda komórka utworzona jest z pewnej liczby bitów.
- Gdy komórkę pamięci tworzy 8 bitów, to mówimy, że *pamięć ma organizację bajtową* — taka organizacja jest typowa dla większości współczesnych komputerów.
- Poszczególne komórki mogą zawierać dane, które są poddawane przetwarzaniu, jak również mogą zawierać rozkazy (instrukcje) dla procesora.

PAMIĘĆ GŁÓWNA (OPERACYJNA) (2)

- Poszczególne bajty (komórki) pamięci są ponumerowane od 0 — numer komórki pamięci nazywany jest jej *adresem fizycznym*.
- Adres fizyczny przekazywany jest przez procesor (lub inne urządzenie) do podzespołów pamięci w celu wskazania położenia bajtu, który ma zostać odczytany lub zapisany.

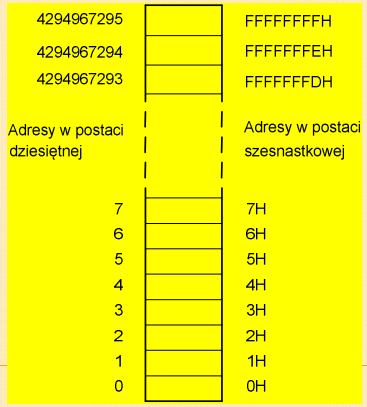
PAMIĘĆ GŁÓWNA (OPERACYJNA) (3)

- W wielu współczesnych procesorach adresy fizyczne są 32-bitowe, co określa od razu maksymalny rozmiar zainstalowanej pamięci głównej (operacyjnej):

  2^{32} = 4 294 967 296 bajtów (4 GiB)

- Poprzez wprowadzenie specjalnych trybów adresowania możliwa jest instalacja i użytkowanie pamięci głównej o rozmiarach 8 GiB, 16 GiB i więcej.

ADRESOWANIE PAMIĘCI GŁÓWNEJ (OPERACYJNEJ) O ROZMIARZE 4 GB



PORZĄDEK BAJTÓW (1)

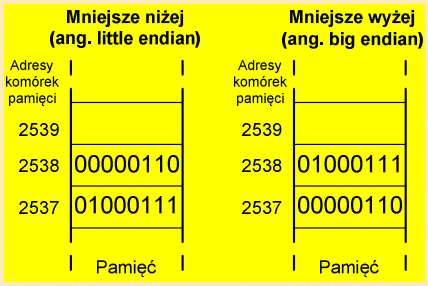
- Pojedynczy bajt (8 bitów) umożliwia zapisanie liczby całkowitej z przedziału <0, 255>, albo jeśli przyjęto stosowanie liczb ze znakiem — z przedziału <–128, +127>
- Ponieważ zapisywane i przetwarzane liczby często przekraczają 255, a w przypadku liczb ze znakiem wychodzą poza zakres <–128, +127>, muszą być więc zapisywane na dwóch, czterech lub na większej liczbie bajtów.
- Pojawia się więc pytanie w jakim porządku bajty powinny być przechowywane lub przesyłane?

PORZĄDEK BAJTÓW (2)

- W systemach komputerowych przyjęto dwa podstawowe schematy określające porządek bajtów:
  - *mniejsze niżej* (ang. little endian)
  - *mniejsze wyżej* (ang. big endian)
- Format *little endian* stosowany jest m.in. w procesorach rodziny x86 (AMD/Intel).
- Format *big endian* stosowany jest m.in. w Internecie i w procesorach Motorola.

PORZĄDEK BAJTÓW (3)

- Przykład zapisu w pamięci liczby $(11001000111)_2$ = $(1607)_{10}$



HIERARCHIA PAMIĘCI (1)

- Ze względu na centralną rolę procesora w pracy komputera, pamięć główna, bezpośrednio współpracująca z procesorem, ma kluczowe znaczenie dla wydajności całego systemu.
- Często zamiast terminu *pamięć główna* używany jest termin *pamięć operacyjna*, a także termin *pamięć RAM* – Random Access Memory, który tłumaczy się jako pamięć o dostępie swobodnym, w której czas odczytu nie zależy od położenia danej w pamięci.

HIERARCHIA PAMIĘCI (2)

- Pamięć główna musi w możliwie najkrótszym czasie przekazywać do procesora żądane rozkazy i dane, jak również zapisywać dane przetworzone przez procesor.
- We współczesnych komputerach instalowana jest pamięć główna wytwarzana w technologii DRAM (ang. Dynamic RAM), w której każdy bit danych przechowywany jest w oddzielnym mikro-kondensatorze. Ten typ pamięci cechuje niski koszt, małe rozmiary i niewielki pobór mocy, wymaga jednak okresowego odświeżania ze względu na rozpraszanie ładunków elektrycznych gromadzonych w kondensatorach.

HIERARCHIA PAMIĘCI (3)

- W poniższej tabeli podano przykładowe czasy dostępu i koszt dla kilku typów pamięci (dane z r. 2008)

Typ pamięci	Typowy czas dostępu	Cena w przeliczeniu na 1 GB
SRAM	0.5 – 2.5 ns	$2000 – $5000
DRAM	50 – 70 ns	$20 – $75
HDD (dyski)	5 000 000 – 20 000 000 ns	$0.20 – $2

- 1 ns = 10^{-9} s

HIERARCHIA PAMIĘCI (3)

- Pamięci SRAM i DRAM są pamięciami ulotnymi, w których zmagazynowane informacje są tracone po wyłączeniu zasilania. Natomiast pamięć dyskowa zachowuje zapisane informacje po wyłączeniu zasilania.
- Ze względu na wysoki koszt, rozmiar pamięci typu SRAM jest ograniczony – w pamięci tej przechowywane są tylko najczęściej używane dane i rozkazy. W komputerach pamięć tego typu nazywana jest pamięcią podręczną (ang. cache memory).

PAMIĘĆ FIZYCZNA I WIRTUALNA (1)

- Rozkazy (instrukcje) programu odczytujące dane z pamięci operacyjnej (czy też zapisujące wyniki) zawierają informacje o położeniu danej w pamięci, czyli zawierają adres danej.
- W wielu komputerach adres ten ma postać ***adresu fizycznego***, czyli wskazuje jednoznacznie komórkę pamięci, gdzie znajduje się potrzebna dana.
- W trakcie operacji odczytu adres fizyczny kierowany do układów pamięci poprzez linie adresowe, a w ślad za tym układy pamięci odczytują i odsyłają potrzebną daną.

PAMIĘĆ FIZYCZNA I WIRTUALNA (2)

- Takie proste adresowanie jest niepraktyczne w systemach wielozadaniowych.
- W rezultacie wieloletniego rozwoju architektury procesorów i systemów operacyjnych wyłoniła się koncepcja ***pamięci wirtualnej***, będącej pewną iluzją pamięci rzeczywistej (fizycznej).
- W przypadku stosowania pamięci wirtualnej aktualnie używane rozkazy i dane przechowywane są w pamięci głównej (operacyjnej) komputera, a pozostałe (tymczasowo niepotrzebne) przechowywane są na dysku.

PAMIĘĆ FIZYCZNA I WIRTUALNA (3)

- Pamięć operacyjna komputera w kształcie widzianym przez programistę nosi nazwę *pamięci wirtualnej*.
- Kompilatory i interpretery języków programowania tworzą rozkazy programu, w których stosowane są adresy wirtualne.
- Transformacja adresów wirtualnych na adresy fizyczne (rzeczywiście istniejących komórek pamięci) jest technicznie dość skomplikowana. Problemy te zostały jednak skutecznie rozwiązane, a związane z tym wydłużenie czasu wykonywania programu zwykle nie przekracza kilku procent.

PAMIĘĆ FIZYCZNA I WIRTUALNA (4)

- W rezultacie w komputerze może być wykonywanych jednocześnie (lub pseudo-jednocześnie) kilka programów wykorzystujących obszary pamięci wirtualnej o tych samych adresach — adresy te są jednak transformowane na adresy w rozłącznych obszarach pamięci fizycznej przydzielonych każdemu programowi.

KODOWANIE ZNAKÓW (1)

- Współczesne komputery posiadają zdolność przechowywania i przetwarzania danych tekstowych — każdy komputer korzysta z pewnego zestawu znaków, do których należą małe i wielkie litery alfabetu łacińskiego, cyfry od 0 do 9, znaki przestankowe, spacja (odstęp międzywyrazowy) i znak nowego wiersza.
- Zestaw ten może być odpowiednio rozszerzany w zależności lokalnych uwarunkowań (np. znaki cyrylicy).

KODOWANIE ZNAKÓW (2)

- Do reprezentacji znaków w komputerze wykorzystuje się przypisane im liczby — odwzorowanie znaków w liczby całkowite tworzy **kod znakowy**.
- Niezbędne jest więc ustalenie sposobów kodowania znaków używanych w tekstach w postaci odpowiednich liczb reprezentowanych w komputerze przez ciągów zer i jedynek.
- Podobny problem kodowania pojawił się kilkadziesiąt lat wcześniej w komunikacji telegraficznej (dalekopisowej) — opracowano wówczas różne schematy kodowania znaków w postaci ciągów zer i jedynek.

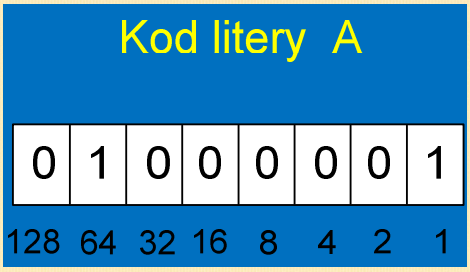
KODOWANIE ZNAKÓW (3)

- W tej sytuacji w systemach komputerowych przyjęto kod opracowany w pierwszej połowie XX w. dla urządzeń dalekopisowych — około roku 1968 w USA ustalił się sposób kodowania znaków znany jako kod ASCII (ang. American Standard Code for Information Interchange).
- Kod ten obejmuje małe i wielkie litery alfabetu łacińskiego, cyfry, znaki przestankowe i sterujące (np. nowa linia).

KOD ASCII (1)

- Znaki w kodzie ASCII zapisywane są na 8 bitach, ale znaki *podstawowego kodu ASCII* (alfabet łaciński, znaki przestankowe i sterujące) wykorzystują tylko kody o wartościach 0 ÷ 127, spośród dopuszczalnego przedziału 0 ÷ 255.
- Podana dalej tablica pokazuje kody ASCII niektórych liter i cyfr.

KOD ASCII (3)



KOD ASCII (2)

- W celu uwypuklenia specyfiki kodu ASCII, wartości liczbowe podane w tabeli zapisane są w systemie szesnastkowym (po liczbie występuje litera H), np. 41H = 4*16 + 1= 65

A	0100 0001	41H	a	0110 0001	61H
B	0100 0010	42H	b	0110 0010	62H
C	0100 0011	43H	c	0110 0011	63H
D	0100 0100	44H	d	0110 0100	64H
E	0100 0101	45H	e	0110 0101	65H
F	0100 0110	46H	f	0110 0110	66H
—	— — — — —		—	— — — — —	
Y	0101 1001	59H	y	0111 1001	79H
Z	0101 1010	5AH	z	0111 1010	7AH

KOD ASCII (4)

0	0011 0000	30H
1	0011 0001	31H
2	0011 0010	32H
3	0011 0011	33H
—	— — — — —	
8	0011 1000	38H
9	0011 1001	39H

!	0010 0001	21H
"	0010 0010	22H
#	0010 0011	23H
$	0010 0100	24H
—	— — — — —	
{	0111 1011	7BH
|	0111 1100	7CH

KOD ASCII (5)

- Pierwotnie, kody ASCII o wartościach od 0 do 31 oraz kod 127 zostały przeznaczone do sterowania komunikacją dalekopisową — niektóre z nich pozostały w informatyce, chociaż zatraciły swoje pierwotne znaczenie, inne zaś są nieużywane.
- Do tej grupy należy m.in. znak powrotu karetki (CR) o kodzie 0DH (dziesiętnie 13) i kod znaku nowego nowej linii (LF) o kodzie 0AH (dziesiętnie 10).

ROZSZERZONE KODY ASCII (1)

- Na bazie *podstawowego kodu ASCII* (kody o wartościach 0 ÷ 127) zaprojektowano wiele *kodów rozszerzonych*, w których wykorzystano także kody o wartościach z przedziału 128 ÷ 255.
- W kodach rozszerzonych pierwsze 128 pozycji jest identyczne, jak w podstawowym kodzie ASCII, a następne 128 pozycji zawiera znaki alfabetów narodowych, symbole matematyczne, itp.

ROZSZERZONE KODY ASCII (2)

- Istnieje wiele kodów rozszerzonych ASCII — tworzą one strony kodowe (ang. code page), czyli 256-elementowe zbiory znaków pewnego języka lub grupy językowej.
- Norma ISO 8859-1 (znana też jako Latin 1) wprowadziła 128 dodatkowych (w stosunku do podstawowego zbioru ASCII) znaków, obejmujących głównie znaki występujące w alfabetach krajów zachodnioeuropejskich.
- Norma ISO 8859-2 zawiera znaki języków Europy Środkowej (czeski, polski, węgierski).

ROZSZERZONE KODY ASCII (3)

- Z kolei norma ISO 8859-3 zawiera znaki języka tureckiego, esperanto i wielu innych.
- w Polsce najbardziej znane są:
  - Windows 1250 (Microsoft CP 1250)
  - ISO 8859–2
  - Latin 2
  - Mazovia (wyszedł z użycia)

ROZSZERZONE KODY ASCII (4)

Znak	a	ą	A	Ą
kody znaków podano w zapisie szesnastkowym				
Latin 2	61	A5	41	A4
Windows 1250	61	B9	41	A5
ISO 8859-2	61	B1	41	A1
Mazovia	61	86	41	8F
Unicode (mniejsze niżej)	61 00	05 01	41 00	04 01
Unicode (mniejsze wyżej)	00 61	01 05	00 41	01 04
UTF-8	61	C4 85	41	C4 84

ROZSZERZONE KODY ASCII (5)

- Korzystanie ze stron kodowych wiąże się z istotnymi problemami:
  - oprogramowanie musi śledzić informacje, która strona jest aktualnie wykorzystywana,
  - nie można łączyć języków z różnych stron kodowych,
  - języka chińskiego i japońskiego nie uwzględniono w stronach kodowych.

UNICODE (1)

- Kodowanie znaków na 8 bitach w postaci stron kodowych okazało się niepraktyczne do przechowywania znaków narodowych krajów europejskich, i tym bardziej niewystarczające dla alfabetów krajów dalekiego wschodu.
- W tej sytuacji okazało się konieczne wprowadzenie kodowania na większej liczbie bitów: 16, a nawet 32 — wówczas wyłonił się standard znany jako **Unicode** (czasami występuje w formie spolonizowanej **Unikod**).

UNICODE (2)

- Podstawą systemu Unicode jest przypisanie każdemu znakowi i symbolowi wartości liczbowej w postaci **punktu kodowego** (ang. code point) — jednolite przypisanie każdemu symbolowi wartości liczbowej upraszcza pisanie oprogramowania.
- Unicode jest definiowany przez dwa standardy: Unicode i ISO 10646 — kody znaków obu standardów są identyczne, a różnice dotyczą drobnych szczegółów. W skład konsorcjum Unicode wchodzą duże firmy komputerowe i producenci oprogramowania, a konsorcjum współpracuje z organizacją ISO.

UNICODE (3)

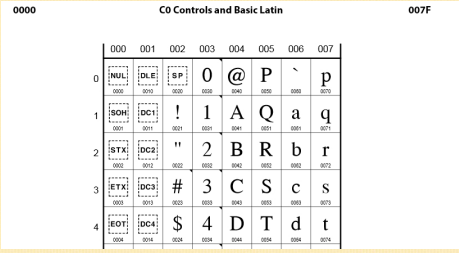
- W zamierzeniu Unicode ma obejmować wszystkie pisma używane na świecie. Przyjmuje się, że w przybliżeniu liczba różnych znaków i symboli, które mogą być zakodowane przekracza milion.
- Zakodowanie ponad miliona różnych znaków w postaci liczb binarnych wymaga stosowania liczb co najmniej 21-bitowych.
- W Unicode przestrzeń punktów kodowych została podzielona na bloki, między innymi alfabet łaciński zajmuje 336 punktów kodowych, grecki 144, cyrylica 256, symbole walut zajmują 48 punktów kodowych, symbole matematyczne 256.

UNICODE (4)

- W odniesieniu do znaków z podstawowego kodu ASCII, w Unikodzie rozszerzono ich kody binarne z 8 do 16 bitów poprzez „dopisanie" 8 zer z lewej strony
- Obok podano przykładowe wartości punktów kodowych w standardzie Unicode – wartości zapisywane są w kodzie szesnastkowym i poprzedzone znakami U+

a U+0061
A U+0041
ą U+0105
Ą U+0104
b U+0062
B U+0042
c U+0063
C U+0043
ć U+0107
Ć U+0106
d U+0064
D U+0044
e U+0065
E U+0045
ę U+0119
Ę U+0118

UNICODE (5)



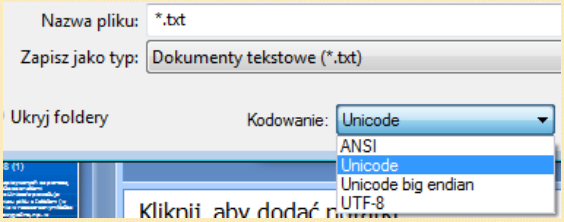
- Fragment oryginalnej tablicy znaków Unicode

KODY BOM (1)

- Ze względu na stosowanie dwóch formatów przechowywania liczb znanych jako *mniejsze niżej / mniejsze wyżej* (ang. little endian / big endian), stosowane są dodatkowe bajty identyfikujące rodzaj kodowania. Bajty te oznaczane są skrótem BOM (ang. Byte Order Mark — znacznik kolejności bajtów) i mają postać (w zapisie szesnastkowym):
  - mniejsze niżej (ang. little endian) FF FE
  - mniejsze wyżej (ang. big endian) FE FF
  - UTF–8 EF BB BF

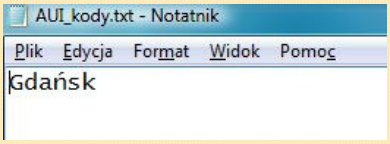
KODY BOM (2)

- W systemie Windows używany jest prosty edytor *Notatnik*. W trakcie zapisywania pliku można określić żądany sposób kodowania tekstu w standardzie Unicode.



KODY BOM (3)

- Przykład: w *Notatniku* napisano wyraz **Gdańsk** i zapamiętano treść pliku przy różnych formatów kodowania. Uzyskana zawartość plików pokazana jest na następnym slajdzie (zawartości poszczególnych bajtów podane są w postaci liczb w zapisie szesnastkowym).



KODY BOM (4)

ASCII (CP 1250)	47	64	61	F1	73	6B								
	G	d	a	ń	s	k								
Unicode (little endian)	FF	FE	47	00	64	00	61	00	44	01	73	00	6B	00
	BOM		G		d		a		ń		s		k	
Unicode (big endian)	FE	FF	00	47	00	64	00	61	01	44	00	73	00	6B
	BOM		G		d		a		ń		s		k	
UTF–8	EF	BB	BF	47	64	61	C5	84	73	6B				
	BOM			G	d	a	ń		s	k				

UTF–8 (1)

- W przypadku tekstów zapisywanych za pomocą alfabetu łacińskiego (także ze znakami narodowymi) stosowanie *Unicode* powoduje co najmniej dwukrotny wzrost rozmiaru pliku z tekstem (w porównaniu do kodowania w rozszerzonym kodzie ASCII), co może być niewygodne, np. w Internecie zwiększa czas przesyłania pliku tekstowego.
- Z tego powodu w ostatnich latach rozpowszechnił się standard kodowania UTF–8 (ang. Unicode Transformation Format), w którym podstawowe znaki kodowane są jako 8-bitowe, a znaki narodowe jako 16-bitowe lub dłuższe

UTF–8 (2)

- W rezultacie tekst zakodowany w formacie UTF–8 jest zazwyczaj o około 5% dłuższy od tekstu w formacie ISO 8859–2.
- Kodowanie w formacie UTF-8 oparte jest na następujących regułach (symbol H podany po liczbie oznacza, że liczba kodowana jest w systemie szesnastkowym):
  - Znaki UCS/Unicode o kodach 00H do 7FH (czyli znaki kodu ASCII) są kodowane jako pojedyncze bajty o wartościach z przedziału 00H do 7FH.

UTF–8 (3)

  - Oznacza to, że pliki zawierające wyłącznie 7-bitowe kody ASCII mają taką samą postać zarówno w kodzie ASCII jak i w UTF–8.
  - Wszystkie znaki o kodach większych od 7FH są kodowane jako sekwencja kilku bajtów, z których każdy ma ustawiony najstarszy bit na 1.

Zakresy kodów	Reprezentacja w postaci UTF-8
00H – 7FH	0*xxxxxxx*
80H – 7FFH	110*xxxxx* 10*xxxxxx*
800H – FFFFH	1110*xxxx* 10*xxxxxx* 10*xxxxxx*
10000H – 1FFFFFH	11110*xxx* 10xxxxxx 10*xxxxxx* 10*xxxxxx*

UTF–8 (4)

- Poniżej podano przykładowe kodowania początkowych liter alfabetu języka polskiego w standardzie UTF–8.

a 61
A 41
ą C4 85
Ą C4 84
b 62
B 42
c 63
C 43
ć C4 87
Ć C4 86
d 64
D 44
e 65
E 45
ę C4 99
Ę C4 98

UTF–8 (5)

- Przykład kodowania litery **ą**
- Litera ą w standardzie Unicode ma przypisany kod 0105H, czyli w postaci binarnej:

0000 0001 0000 0101

- Ponieważ kod 0105H należy do przedziału <80H – 7FFH>, więc litera ą będzie kodowana na dwóch bajtach postaci:

110*xxxxx* 10*xxxxxx*

UTF–8 (6)

- Z podanego kodu bierzemy pod uwagę ostatnie 11 bitów (zaznaczone kolorem czerwonym) i wpisujemy w miejsca oznaczone ***xxxxx*** i ***xxxxxx***.

110*00100* 10*000101*

- Obliczona wartość po zamianie na system szesnastkowy ma postać C4H 85H, co jest zgodne z wcześniej podaną tabelą.

UTF–16 (1)

- Kodowanie UTF–16 przeznaczone jest do reprezentacji znaków Unikodu w środowiskach lub kontekstach ukierunkowanych na słowa 16-bitowe.

	kod UTF-16
a	0061
A	0041
ą	0105
Ą	0104
b	0062
B	0042

W przypadku znaków z grupy BMP (kody od 0H do FFFFH), kod UTF–16 jest identyczny z wartością punktu kodowego. Obok podano przykładowe kody kilku liter alfabetu języka polskiego w formacie UTF-16 (zapis szesnastkowy).

UTF-16 (2)

- Dla znaków z przedziału od 10000H do 10FFFFH (nie należących do BMP) stosuje się dwa słowa 16-bitowe.
- Wartość z przedziału od 10000H do 10FFFFH zostaje najpierw pomniejszona o 10000H.
- W rezultacie pojawia się wartość 20-bitowa, z której 10 najstarszych bitów wpisywana jest do pierwszego słowa (pole *xxxxxxxxxx*), a pozostałe 10 bitów wpisywanych jest do drugiego słowa (pole *yyyyyyyyyy*), tak jak pokazano poniżej.

110110 *xxxxxxxxxx*	**110111** *yyyyyyyyyy*

UTF-16 (3)

- Przykładowo, symbolowi CAT (kot) przyporządkowano wartość punktu kodowego 1F408 (zazwyczaj podawaną w postaci U+1F408).
- W celu uzyskania kodu UTF-16 odejmujemy najpierw liczbę 10000 (liczby w zapisie szesnastkowym i dwójkowym)

$$(1F408)_{16} - (10000)_{16} = (0F408)_{16} =$$
$$= (0000111101\ 0000001000)_2$$

UTF-16 (4)

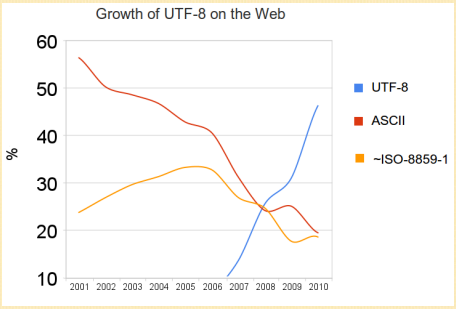
- Starsze 10 bitów poprzedzamy ciągiem $(110110)_2$ i otrzymujemy ciąg 16-bitowy:

$$1101\ 1000\ 0011\ 1101 = (D83D)_{16}$$

- Z kolei młodsze 10 bitów poprzedzamy ciągiem $(110111)_2$ i otrzymujemy ciąg 16-bitowy:

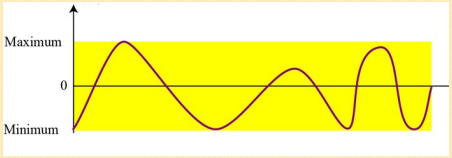
$$(1101\ 1100\ 0000\ 1000)_2 = (DC08)_{16}$$

ROZPOWSZECHNIENIE RÓŻNYCH STANDARDÓW KODOWANIA ZNAKÓW



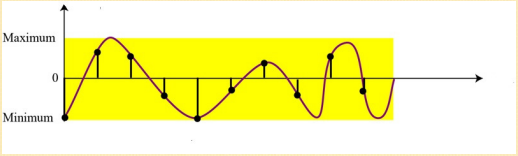
REPREZENTACJA DŹWIĘKU (1)

- Dźwięki z natury są całkowicie odmienne od liczb lub tekstów – tekst składa się ze znaków, które są policzalne. W przeciwieństwie do tego, sygnały dźwiękowe nie są policzalne. Gdyby możliwe było zmierzenie wartości tych sygnałów, to trzeba by je zapisać w pamięci o nieskończonej pojemności.



REPREZENTACJA DŹWIĘKU (2)

- Rozwiązaniem problemu jest próbkowanie czy rejestrowanie wartości sygnału co pewien czas.



REPREZENTACJA DŹWIĘKU (3)

- Przyjmuje się, że urządzenia elektroakustyczne powinny przesyłać dźwięki o częstotliwościach do około 20 kHz. Kierując się tymi zasadami w zwykłych odtwarzaczach CD przyjęto częstotliwość próbkowania 44100 Hz. Również karty dźwiękowe w komputerach są przystosowane do próbkowania z tą częstotliwością.
- Niekiedy wybór częstotliwości próbkowania należy do użytkownika. Wybór ten stanowi kompromis między wielkością pamięci (czy pliku) a wymaganiami dotyczącymi jakości odtwarzania.
-

REPREZENTACJA DŹWIĘKU (4)

- W zastosowaniach związanych z rejestracją mowy częstotliwość próbkowania 6 kHz można uważać za minimalną.
- Przykładowo, rejestracja muzyki z częstotliwością próbkowania 44100 Hz przez minutę, przy założeniu, że każda próbka zajmuje jeden bajt, powoduje zapisanie 44100 * 60 = 2 646 000 bajtów.

REPREZENTACJA DŹWIĘKU (5)

- W sygnałach cyfrowych mamy także do czynienia z kwantyzacją amplitudy. W urządzeniach do przesyłania i przetwarzania sygnałów dźwiękowych próbki rejestrowane w postaci kodu 8- lub 16-bitowego.
- Oznacza to, że cały zakres dynamiki dźwięku może być reprezentowany przez 256 lub 65536 poziomów wartości. Kodowanie 16-bitowe, stosowane m.in. przez odtwarzacze CD, zapewnia znacznie wyższą jakość odtwarzanych dźwięków. Niektórzy autorzy piszą, że dźwięk o rozdzielczości 8-bitowej jest odbierany jako płaski i hałaśliwy.

MIME (1)

- MIME (ang. Multipurpose Internet Mail Extension) — standard pozwalający przesyłać w sieci Internet wszelkie dane (teksty, grafikę, zdjęcia, dźwięki, itp.).
- Zastosowanie MIME (wbrew nazwie) nie ogranicza się tylko do plików przesyłanych pocztą.

MIME (2)

- Jeśli kliknie się na stronie WWW łącze (link), który wskazuje na jakiś plik, to wówczas serwer strony przekazuje przeglądarce typ MIME tego pliku, np. text/html jest typem przypisanym do zwykłej strony WWW.
- W konfiguracji przeglądarki każdy typ skojarzony jest z określonym programem, który należy uruchomić po ściągnięciu tego typu pliku, bądź działaniem, które powinna wykonać sama przeglądarka (np. zapis pliku na dysk).

MIME (3)

- Obsługa typowych formatów realizowana jest przez samą przeglądarkę, pozostałe powodują uruchomienie specyficznego programu lub propozycję zapisu pliku na dysku.
- Typ MIME składa się z dwóch elementów oddzielonych ukośnikiem:
  - typu pliku (na przykład obraz (image), dźwięk (audio) lub tekst (text)),
  - podtypu pliku, który definiuje jego konkretny format (na przykład jpeg, gif, html).

MIME (4)

- Przykładowo, obrazek w formacie JPEG, posiada określony typ MIME jako "image/jpeg" lub "image/pjpeg" (w przypadku pliku z kompresją progresywną).
- Podstawowe typy danych są następujące:
  - text/plain tekst (niesformatowany) w kodzie ASCII
  - text/html strona w formacie HTML
  - image/gif obrazek w formacie GIF
  - image/jpeg obrazek w formacie JPG
  - application/postscript tekst sformatowany i grafika PS
  - audio/basic format dźwiękowy 8-bitowy (jednokanałowy strumień audio)
  - video/mpeg animacja MPEG

MIME (5)

- Zazwyczaj, w nagłówku przesyłanej informacji pojawia się numer standardu (1.0). Dalej występuje opis zawartości pliku oraz, jeśli jest to plik tekstowy, zestaw znaków stosowanych w tym pliku. Przykładowy nagłówek może mieć postać:

  MIME-Version: 1.0
  Content-Type: text/plain;charset=iso-8859-2
  Content-Transfer-Encoding: 8bit

- Jako zestaw dość często występuje także charset=UTF-8

MIME (6)

- Początkowo, w sieci Internet możliwe było tylko przesyłanie znaków o kodach 0 ÷ 127, dlatego wiele programów nadal w trakcie przesyłania kodów o wartościach powyżej 127 korzysta ze specjalnych technik. Informacja o sposobie kodowania zawarta jest w nagłówku MIME w jednej z postaci:
  - Content-Transfer-Encoding: quoted-printable
  - Content-Transfer-Encoding: base64
  - Content-Transfer-Encoding: 8bit *(dane nie wymagają żadnej konwersji)*.

KODOWANIE *QUOTED-PRINTABLE* (1)

- Sposób *quoted-printable* stosuje się, gdy dane zawierają niewiele znaków spoza standardowego ASCII, np. do przesyłania tekstów z literami specyficznymi dla alfabetu języka polskiego.
- Znaki o kodach większych od 127 zamieniane są na trzy znaki z podstawowego zestawu ASCII:
  - znak =
  - dwie cyfry w zapisie szesnastkowym określające kod znaku
- Znak równości jest zapisywany jako =3D

KODOWANIE *QUOTED-PRINTABLE* (2)

- Przykładowo, kod litery **ę** w standardzie ISO-8859-2 ma wartość 234 (lub EA w zapisie szesnastkowym).
- Litera będzie więc reprezentowana przez trzy znaki: **=EA**

KODOWANIE *BASE64* (1)

- Kodowanie *base64* stosowane jest często do kodowania plików binarnych zawierających dane inne niż tekstowe (np. grafika, dźwięk).
- W standardzie *base64* używana jest pokazana niżej tablica kodująca, która przyporządkowuje dla każdej liczby z przedziału <0, 63> odpowiedni znak w kodzie ASCII.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

52	53	54	55	56	57	58	59	60	61	62	63
0	1	2	3	4	5	6	7	8	9	+	/

KODOWANIE *BASE64* (2)

- W wyniku kodowania w standardzie *base64* uzyskuje się kod zawierający wyłącznie znaki ASCII (podane w tablicy na poprzedniej stronie), które mogą być przesyłane jak zwykły tekst.
- W standardzie *base64* trzy kolejne bajty pliku traktuje się jako ciąg 24 bitów, które dzieli na cztery grupy po 6 bitów.
- Zawartość grupy 6-bitowej traktuje się jako liczbę binarną z przedziału 0 ÷ 63. Kodowanie polega na przypisaniu każdej grupie 6-bitowej odpowiedniego znaku z tablicy.

KODOWANIE *BASE64* (3)

Przykładowo, jeśli początkowe bajty pliku mają postać: 219, 165, 45, to po przekodowaniu otrzymamy ciąg znaków ASCII: "26Ut".

219 (DBH)								165 (A5H)								45 (2DH)							
1	1	0	1	1	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	1	1	0	1
1	1	0	1	1	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	1	1	0	1
54								58				20				45							
"2"								"6"				"U"				"t"							