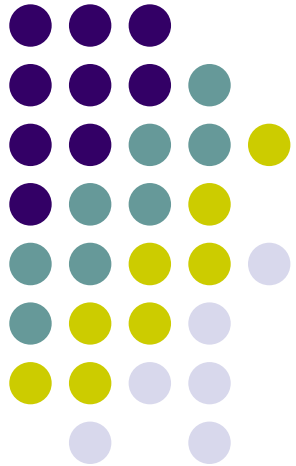
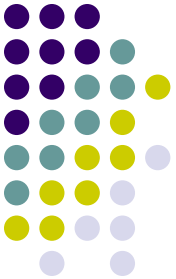
Architektura počítačů

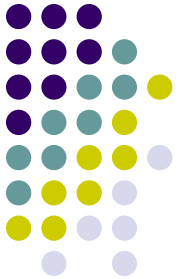


Co je to počítač ?



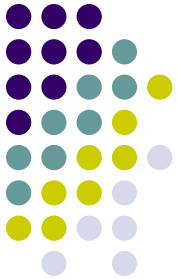
- **Počítač** je zařízení, které podle předem připraveného **programu** zpracovává **data**
- **Data** = informace (čísla, text, zvuk, obraz, výsledky měření, signály z čidel…)
- **Program** = posloupnost instrukcí
- **Instrukce** = jednorázový povel, který umí počítač vykonat

Co je to počítač



- Pod pojmem počítač si mnoho lidí představí PC s klávesnicí, monitorem, případně notebook
- Počítač je ale obecně jakékoliv zařízení, které pracuje s daty podle zadaného programu
- Počítačem je tedy například i váš mobilní telefon – má paměť (dokonce několik pamětí různého typu), vstupní periferie (klávesnice, anténa, mikrofon), výstupní periferie (displej, anténa, reproduktor) a je naprogramován – jeho činnost je popsána tzv. firmwarem
- Počítačem je i váš MP3 přehravač, digitální fotoaparát, dvd přehravač…

Co je to počítač ?

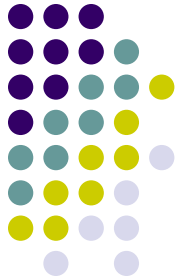


- Existují dva základní typy počítačů:
  - **Analogový počítač -** zpracovává analogová data
  - **Číslicový počítač -** zpracovává digitální data

- **Analogové počítače** bývají úzce specializované obvykle na jednu úlohu nebo pouze na jednu třídu úloh (např. řešení diferenciálních rovnic) a chci-li řešit jiný typ úlohy, musím počítač přestavět

- **Číslicové počítače** lze snadno zkonstruovat jako univerzální

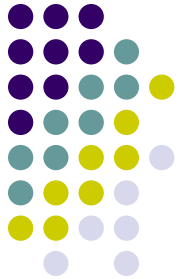
Analogový X Číslicový



- **Analogový počítač** zpracovává analogový signál – informace je reprezentována úrovní signálu
- Základním stavebním prvkem je **operační zesilovač**

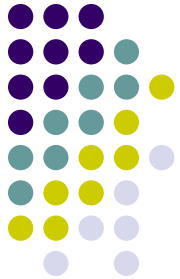
- **Číslicový počítač** zpracovává číslicový (digitální) signál – signál nabývá pouze hodnot 0 nebo 1
- Základním stavebním prvkem je **logický obvod**

Logické obvody



- Zopakovat
  - Negace
  - AND
  - OR
  - NAND
  - NOR
  - XOR
- NAND jako univerzální stavební prvek
- Kombinační x sekvenční obvod
- JK obvod

Co je to vlastně ten signál ?



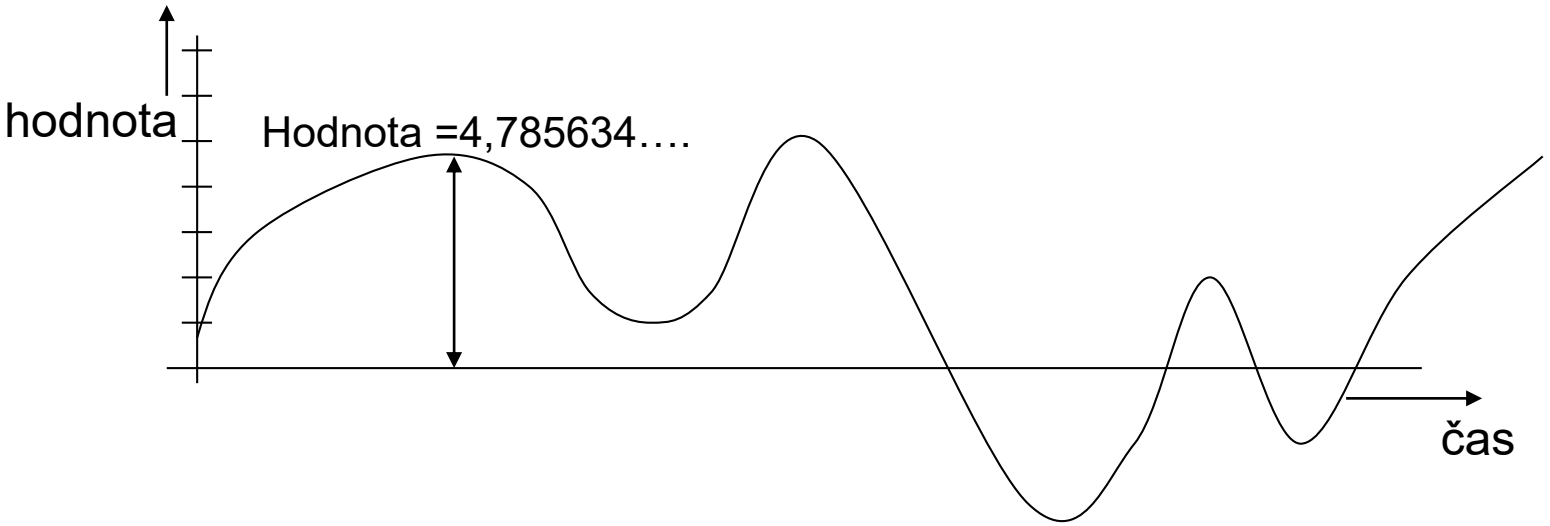
- Za signál považujeme fyz. veličinu měnící se v čase nebo v prostoru a nesoucí obvykle nějakou informaci

- Přesná definice by mohla znít: Signál je veličina měnící svou hodnotu v časoprostoru

- V elektrotechnice obvykle zkoumáme průběh hodnoty napětí v závislosti na čase

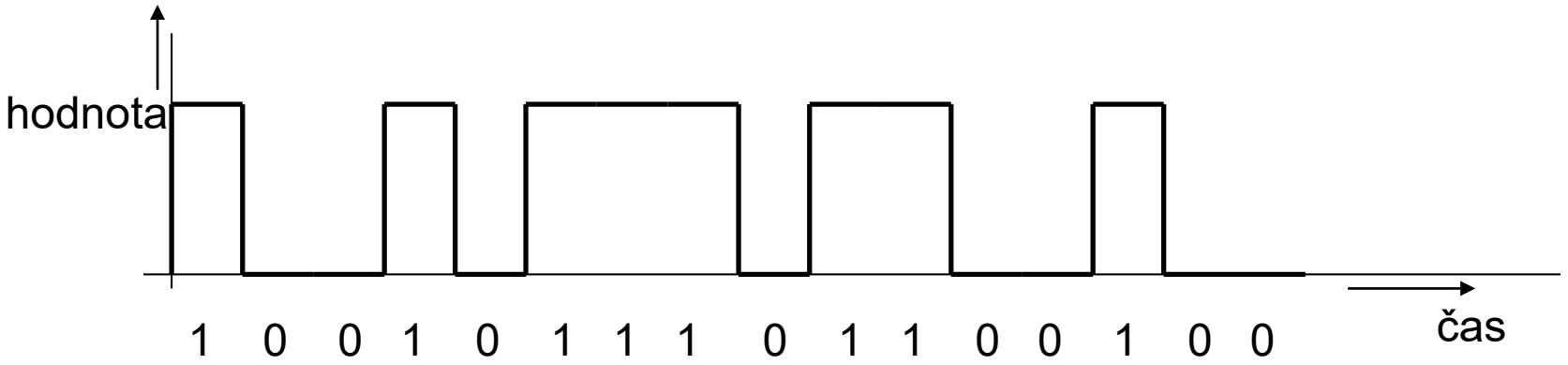
Analogový signál

- Analogový signál je **spojitý** v čase i v oboru hodnot
- V jakýkoliv okamžik známe přesnou hodnotu analogového signálu
- Informace je v analogovém signálu zakódována okamžitou přesnou hodnotou signálu (např. výchylka membrány reproduktoru odpovídá hodnotě zvukového signálu a ta se mění v čase tak, jak se mění akustický tlak působící na mikrofon)



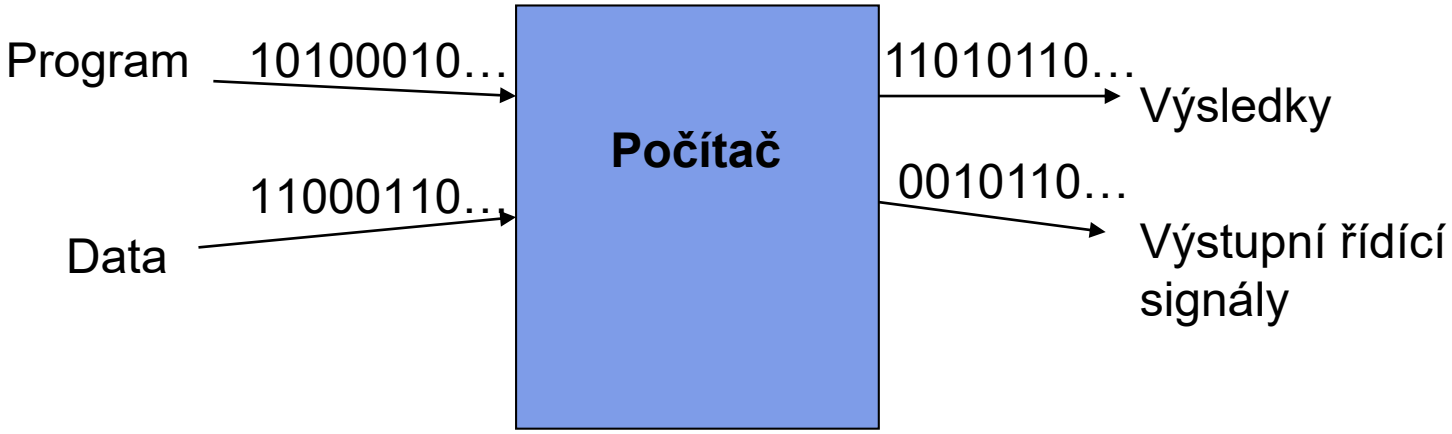
Digitální signál

- Digitální signál je **nespojitý** v čase i v hodnotách
- Hodnota digitálního signálu může nabývat pouze dvou různých úrovní (logická 0 nebo 1) a mění se skokově v pravidelných intervalech
- Informace je v digitálním signálu zakódována kombinací jedniček a nul
- Musí tedy existovat nějaký **kód**, který informace z reálného světa převádí na **číslicovou podobu** (kombinaci bitů)

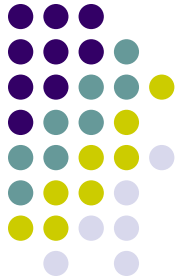


Číslicové počítače

- V tomto předmětu se dále budeme věnovat pouze číslicovým počítačům
- Číslicový počítač pracuje s číslicově kódovanou informací – tedy informací reprezentovanou pomocí **bitů**
- Veškerá zpracovávaná **data**, ale i **program**, dle kterého počítač pracuje, mají číslicovou podobu (kombinace bitů)

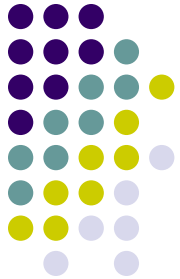


Bit



- **Bit** je základní a současně nejmenší jednotkou informace
- Značí se malým písmenem ***b***
  - Například 24 b
- 1 bit reprezentuje informaci, získanou odpovědí na jednu otázku typu ano/ne, u které je pravděpodobnost obou odpovědí stejná
- Stav bitu – bit může nabývat pouze dvou možných logických stavů, pro jejichž vyjádření používáme číslice 1 a 0 nebo matematické pojmy pravda/nepravda (True/False)
- Stav bitu
  - 0 – Nepravda - False
  - 1 – Pravda - True

Bity



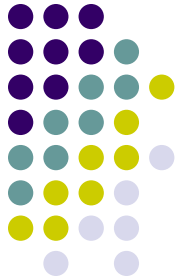
- Složitější informace lze zakódovat pomocí **kombinace více bitů**
- Pomocí **n** bitů lze vytvořit $\mathbf{2^n}$ kombinací jejich stavů

- Například pomocí 2 bitů lze vytvořit 4 dvoubitové kombinace
  (00 , 01 , 10 , 11)

- Pomocí 3 bitů lze vytvořit 8 kombinací
  (000, 001, 010, 011, 100, 101, 110 , 111)

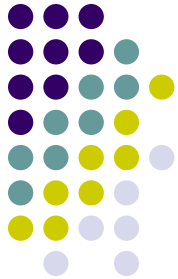
- Jednotlivým bitovým kombinacím lze pomocí zvoleného **kódu** přiřadit nějaký reálný význam
- Například
  - 00-jaro
  - 01-léto
  - 10-podzim
  - 11-zima
- Vidíme tedy, že informace o ročním období je dvoubitová

Kódy



- Existuje celá řada kódů, které převádí číselné, textové ale i libovolné jiné informace do číslicové podoby
- Informace v číslicové podobě (tzn. zakódovaná pomocí určité kombinace bitů) může být zpracovávána číslicovým počítačem

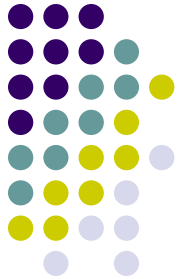
Binární kód



- Nejjednodušší kód pro převod číselné hodnoty do digitální podoby
- Využívá se **dvojková (binární) soustava**
- Ve dvojkové soustavě se pro zápis hodnoty čísla používají pouze číslice **0** a **1**
- Číslice **0** a **1** se zároveň používají pro vyjádření **stavu bitu**
- Zápisem čísla ve dvojkové soustavě tedy vlastně získáváme jeho reprezentaci pomocí bitů

- Příklad:
- Hodnota 179
- Číslo převedeme do dvojkové soustavy: 179 = $(10110011)_2$
- Hodnota 179 se tedy dá zakódovat pomocí **8 bitů** a této hodnotě odpovídá kombinace 10110011

Opakování



- Mocniny dvojky (2^0 až 2^8, 2^{10}, 2^{16}, 2^{20}, 2^{24}, 2^{30}, 2^{32}, odvození $2^{22}=2^{20} \times 2^2$…)
- Převod z binární hodnoty na dekadickou
- Převod dekadické hodnoty na binární
  - Pomocí dělení a zápisu zbytků
  - Postupnou aproximací (odečítání mocnin dvojky)
- Převod desetinné části reálného čísla (BIN>DEC, DEC>BIN)

Reálná čísla

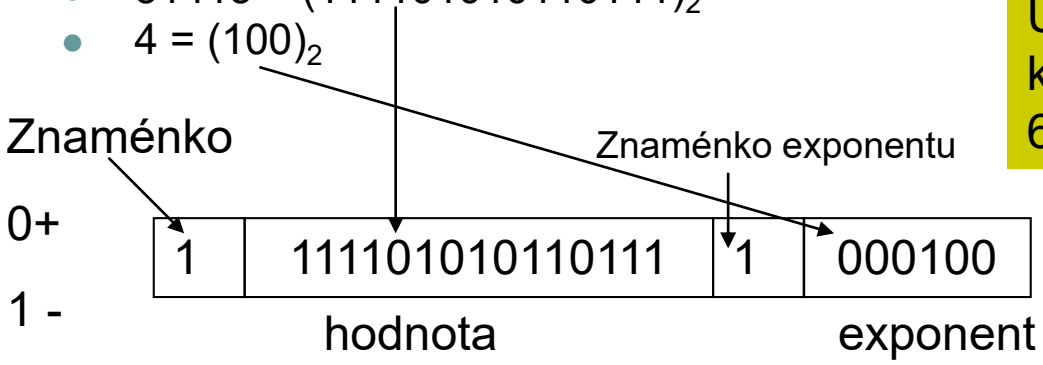
- Reálná čísla mohou mít záporné **znaménko** a **desetinnou část**.
- K zakódování takových č m
  převod do dvojkové sous
- Existuje celá řada kódů p
- Většina těchto kódů použ
  - 1 bit pro zakódování
  - N bitů pro zakódován
  - 1 bit pro zakódování
  - M bitů pro zakódování hodnoty exponentu

Poznámka:

Uvedený kód je opravdu pouze ukázkový.

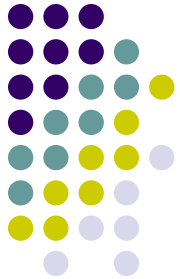
Skutečně používané kódy používají mnohem více bitů a pro exponent se nepoužívá mocnina desítky, ale mocnina dvojky

- Příklad:
  - Hodnota -3,1415 = -31415 x 10^{-4} (zbavili jsme se desetinné části)
  - 31415 = $(111101010110111)_2$
  - 4 = $(100)_2$



Ukázkový jednoduchý 24-bitový kód používá 16 bitů pro hodnotu a 6 bitů pro exponent.

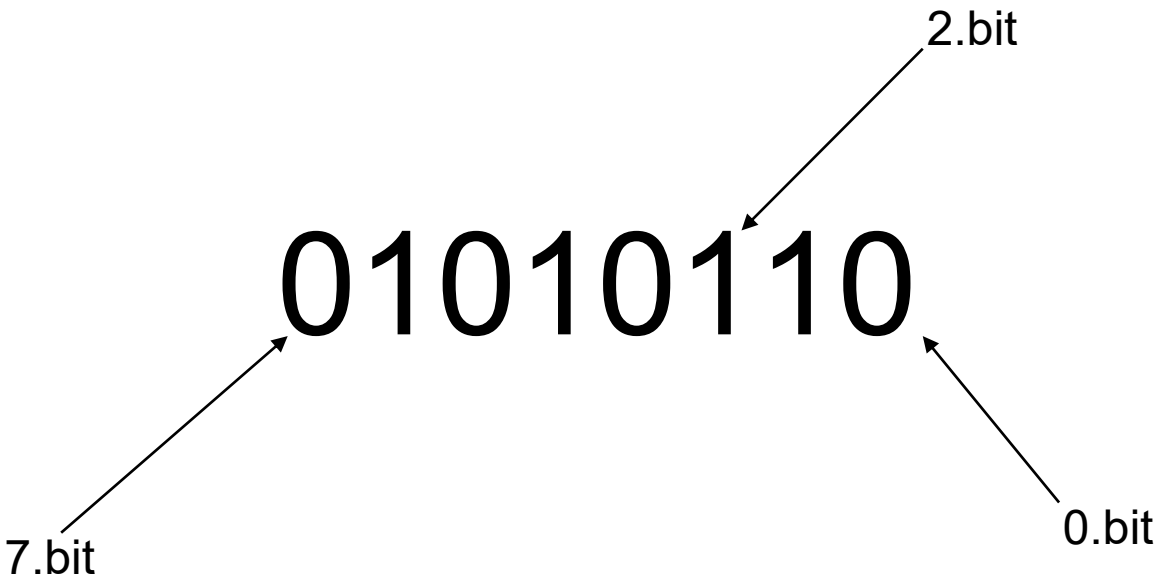
Bajt



- Počítače, mikroprocesory a zařízení výpočetní techniky obvykle nepracují s jednotlivými bity, ale zpracovávají data ve větších **kvantech** (porcích) naráz
- Základní porcí, ve kterých jsou data v informatice zpracovávána, je skupina osmi bitů zvaná **Bajt** – anglicky **Byte**
- Slovo ***byte*** pochází z anglického slova ***bite*** (*sousto*, tzn. nejmenší objem dat, s kterým počítač dokáže manipulovat), které bylo upraveno (i→y), aby se předešlo záměně se slovem ***bit***

- Termín **byte** zavedl Werner Buchholz v roce 1956, při práci na počítači IBM Stretch
- Zpočátku tento termín popisoval skupinu 1-6 bitů
- Později se osmibitový byte stal standardem pro počítač IBM System/360 a jeho popularita pak vedla k tomu, že osmibitový bajt je dnes standardem

Bajt

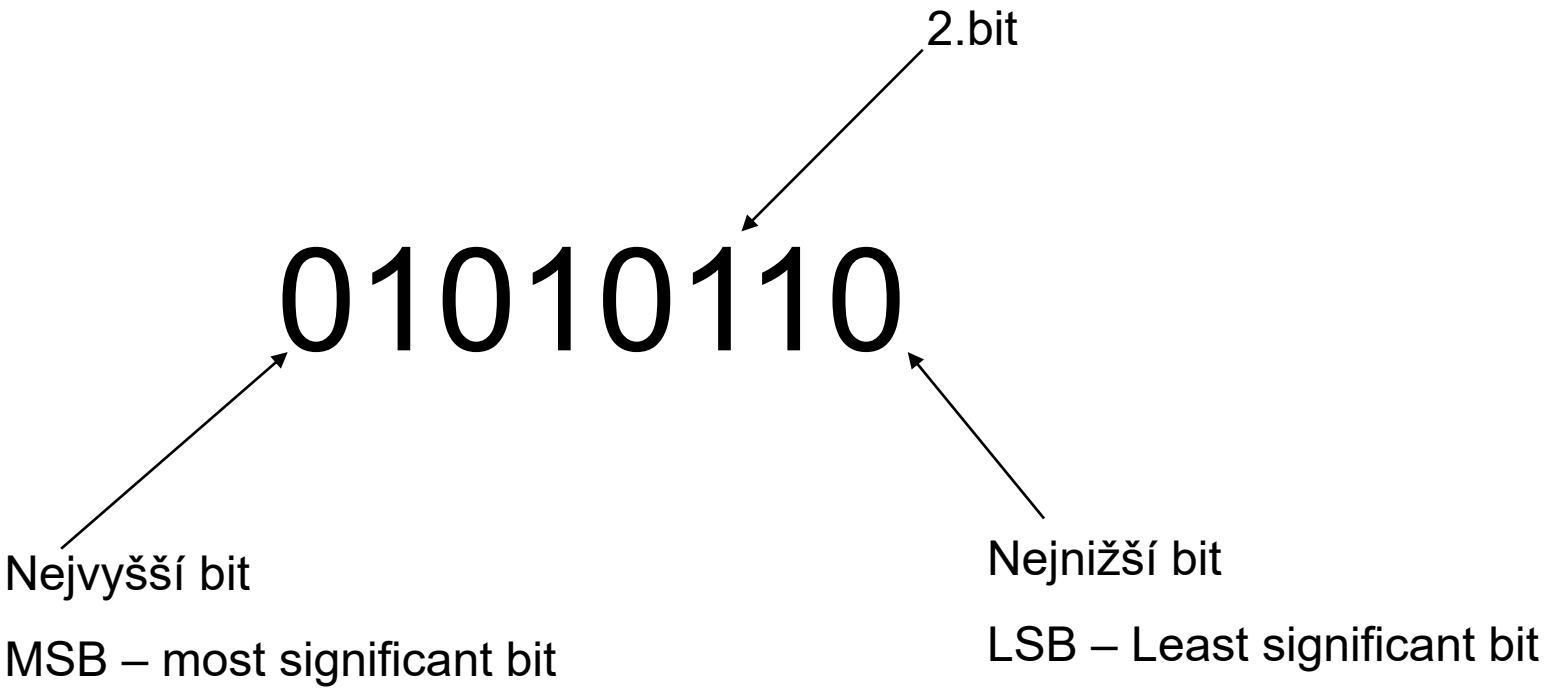


Pozor na pořádová čísla jednotlivých bitů !

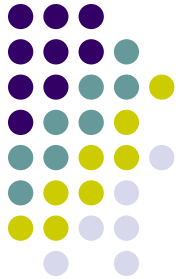
Bajt má **8 bitů**

Bajt má **nultý** až **sedmý** bit

Bajt


2.bit
01010110
Nejvyšší bit
MSB – most significant bit
Nejnižší bit
LSB – Least significant bit

Bajt



- **Stav bajtu** lze vyjádřit třemi typickými způsoby

- 8-bitová kombinace

  11000101

  Bajt je osmice bitů, každý bit má stav 0/1. Bajt lze tedy vyjádřit jako kombinaci stavů osmi bitů

- Hodnota

  197

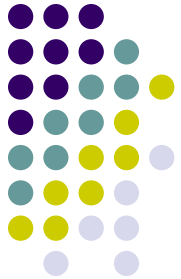
  Osm bitů lze chápat jako osmiciferné binární číslo. Takové číslo může nabývat hodnot 0-255 (00000000b až 11111111b)

- Hexadecimální hodnota

  C5

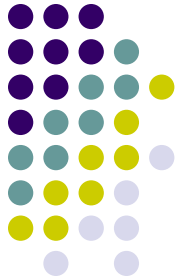
  Tento způsob je nejpoužívanější, protože je nejúspornější (jakýkoliv stav bajtů lze zapsat pomocí dvou cifer) a snadno vzájemně převeditelný na kombinaci osmi bitů (4 bity = 1 hex. cifra)

Bajty a bity



- **Bajt** označujeme písmenkem velké **B**
- Pozor na záměnu **b** a **B** (bit, Bajt)
- Pro co nejvyšší jednoznačnost se proto bit někdy nezkracuje a ponechává se jako „**bit**“ (např. **Mbit/s** pro megabit za sekundu)
- S bajty i bity se používají běžné předpony soustavy SI jako kilo-, mega-, giga-
- například 23 GB, 8 Mb/s
- Tyto předpony však mají obvykle odlišný význam, který je nutno rozlišit z kontextu
- Z technologických důvodů jsou velikosti polovodičových počítačových pamětí rovny nějaké mocnině dvou (1024 B, 2048 B, 4096 B, 8192 B …..)
- Polovodičová paměť nikdy nebude mít kapacitu 1000 bajtů, ale vždy bude vyrobena s kapacitou 1024 bajtů (proč tomu tak je, se dozvíme později)
- **$1024 = 2^{10}$**
- Z tohoto důvodu byl v informatice od počátku předponě **kilo-** přiřazen násobek **1024x**
- Předpona **mega-** je potom násobek **2^{20}** tedy 1024x1024, což je 1048576
- Tyto upravené předpony se používají při vyjadřování velikosti **polovodičových pamětí**, zatímco například kapacity **pevných disků a dalších úložišť** používají dekadické předpony. Bez znalosti kontextu tedy může být složité určit, jaký význam byl zamýšlen, přičemž rozdíl může dosahovat až několika procent

Kilo, Mega, Giga, Tera….



- 1 MB paměti RAM je 1048576 bajtů ← Správně by mělo být 1 MiB
- 1 MB kapacity disku je 1000000 bajtů

- nejasnosti kolem skutečné kapacity prodávaných pamětí a disků vedly k právním sporům
- V prosinci 1998 proto IEC vytvořila dodatek k normě IEC 60027-2, ve kterém zavedla pro počítačové jednotky nový systém označování násobků.
- V tomto systému bylo pro původní „velké kilo“ = 1024 B navrženo označení **kibibyte** a značka **KiB**
- Zatímco jednotka **kilobyte** (se značkou **kB**) označuje **1000 B**, tak jak je obvyklé v soustavě SI
- To ale nebyl šťastný krok….
- Nové předpony **Ki**, **Mi** se příliš neujaly
- Stará literatura a dokumenty navíc používají pouze **kB**, **MB**, **GB** a v drtivé většině případů je tím myšlen násobek 1024 (tedy pravý opak nového významu této předpony)
- Chaos se tedy bohužel ještě prohloubil

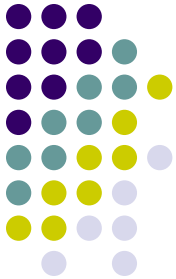
Význam předpon

Ki	Kibi	2^{10}	1024	
Mi	Mebi	2^{20}	1024*1024	=1048576
Gi	Gibi	2^{30}	1024*1024*1024	=1073741824
Ti	Tebi	2^{40}	1024*1024*1024*1024	=1099511627776

K	Kilo	10^{3}	1000	
M	Mega	10^{6}	1000*1000	=milion
G	Giga	10^{9}	1000*1000*1000	=miliarda
T	Tera	10^{12}	1000*1000*1000*1000	=bilion

Tato čísla nám připadají „kulatá“ a hezká, ale to je jen iluze díky desítkové soustavě. Používání desítkové soustavy, ale nemá žádný rozumný důvod a smysl. Jediný důvod, proč v ní počítáme je ten, že jsme na ní od narození zvyklý a máme deset prstů. Ve skutečnosti jde z matematického hlediska o bezvýznamné hodnoty.

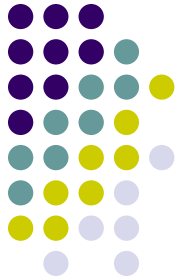
Význam předpon vyjádřený v soustavách se základem 2^n



Ki	Kibi	2^{10}	$(10000000000)_2$ = 400 h
Mi	Mebi	2^{20}	$(10000000000000000000)_2$=100000 h
Gi	Gibi	2^{30}	$(1000000000000000000000000000000)_2$=40000000 h
Ti	Tebi	2^{40}	$(10000000000000000000000000000000000000000)_2$=10000000000h

K	Kilo	10^3	$(1111101000)_2$ = 3E8 h
M	Mega	10^6	$(11110100001001000000)_2$ = F4240 h
G	Giga	10^9	$(11101110011010111001010000000000)_2$ = 3B9ACA00 h
T	Tera	10^{12}	$(11101000110101001010010100010000000000000)_2$ = E8D4A51000 h

V soustavách se základem 2n již čísla milion, milarda a bilion nevypadají nijak sympaticky a kulatě. Naopak se ukazuje, že ve skutečnosti jsou daleko „hezčí a kulatější“ čísla vycházející z mocnin dvojky – tato čísla mají také mnohem větší matematický význam v mnoha aplikacích

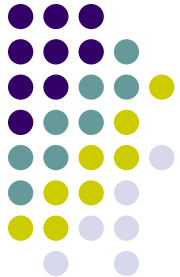


Příklad

- Nově zakoupený pevný disk má výrobcem udávanou kapacitu **1 TB**. Kolik volného místa na prázdném disku zobrazí OS Windows?

- 1 TB = 1 000 000 000 000 B dle výrobců disků
- OS Windows zobrazuje kapacitu paměti, ale i velikost souborů na disku a volné místo na disku v KiB, MiB, GiB a TiB, ale nesprávně je označuje jako KB, MB, GB a TB (takhle to ale v podstatě dělá dnes každý, novou normu téměř všichni výrobci softwaru a hardwaru ignorují)
- 1 TB = (**1 000 000 000 000** / 1024) KiB = 976 562 500 KiB
- 1 TB = (976562500 / 1024) MiB = 953 674 MiB
- 1 TB = (953674 / 1024) GiB = 931 GiB
- OS Windows nám ukáží, že kapacita pevného disku **931 GB** (správně by to mělo být zobrazeno jako 931 **GiB**)

ASCII kód



- *American Standard Code for Information Interchange* (americký standardní kód pro výměnu informací)
- Nejpoužívanější kód pro převod **textu** (tedy písmen a znaků) do číslicové podoby
- **ASCII kód** je definován **tabulkou**
- Pro každý běžný zobrazitelný znak (písmena, číslice, interpunkční znaménka, běžné symboly) přiřazuje **osm bitů**
- ASCII kód tedy zakóduje **jeden znak** jako **jeden bajt**
- ASCII tabulka obsahuje 256 různých znaků
- Prvních 128 znaků je dáno jednoznačně
- Dalších 128 znaků (zakódováno bajty s hodnotou 128-255) není pevně dáno a může se lišit podle národního prostředí (různá kódování českých znaků apod.)

- Prvních 32 znaků (bajty 0 až 31) je „neviditelných“ – jde o speciální kódy pro nezobrazitelné znaky typu konec řádku, pípnutí, posun atd.
- Tyto neviditelné znaky se uplatňovaly například při ovládání starých tiskáren, které tiskly pouze ve znakovém režimu, nebo při přenosu znaků mezi vzdáleným terminálem

ASCII tabulka

- Ukázka části ASCII tabulky

Bajty s hodnotou 128 -255 mohou v různých regionech různý význam (sloupečky IBM, ISO, Win, Kam odpovídají různým variantám kódování speciálních znaků)

Dec	Hex	IBM	Dec	Hex	IBM	Dec	Hex	IBM	ISO	Win	Ka
0	0		64	40	@	128	80	Ç		Č	Ç
1	1	☺	65	41	A	129	81	ü		ü	ü
2	2	☻	66	42	B	130	82	é	‚	é	é
3	3	♥	67	43	C	131	83	â		ď	â
4	4	♦	68	44	D	132	84	ä	„	ä	ä
5	5	♣	69	45	E	133	85	à	…	Ď	ů
6	6	♠	70	46	F	134	86	å	†	Ť	ć
7	7	•	71	47	G	135	87	ç	‡	č	ç
8	8	◘	72	48	H	136	88	ê	ˆ	ě	ł
9	9	○	73	49	I	137	89	ë	‰	Ě	ë
10	A	◙	74	4A	J	138	8A	è	Š	Ĺ	Ő
11	B	♂	75	4B	K	139	8B	ï	‹	Í	ő
12	C	♀	76	4C	L	140	8C	î	Ś	ˇ	î
13	D	♪	77	4D	M	141	8D	ì	Ť	Í	Ź
14	E	♫	78	4E	N	142	8E	Ä	Ž	Ä	Ä
15	F	☼	79	4F	O	143	8F	Å	Ź	Á	Ć
192	C0	└								Ŕ	Ŕ
193	C1	┴								Á	Á
194	C2	┬								Â	Â
195	C3	├								Ă	Ă
196	C4	─								Ä	Ä
204	CC	╠								Ě	É
205	CD	═								Í	Í
206	CE	╬								Î	Î
207	CF	╧								Ď	Ď

AHOJ lze zakódovat pomocí **4 bajtů** s hodnotami 65, 72, 79, 74

tj. *hexadecimálně:* 41, 48, 4F, 4A

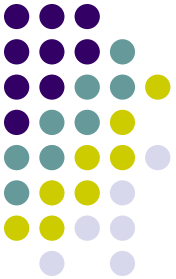
tj. *dvaatřicetibitově:*

01000001010010000100111101001010

DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII
1	☺	32	space	64	@	96	`	128	Ç	160	á	192	└	224	Ó
2	☻	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
3	♥	34	"	66	B	98	b	130	è	162	ó	194	┬	226	Ô
4	♦	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
5	♣	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
6	♠	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
7	•	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
8	◘	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
9	○	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
10	◙	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
11	♂	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
12	♀	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
13	♪	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
14	♫	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
15	☼	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
16	►	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
17	◄	48	0	80	P	112	p	144	È	176	░	208	ð	240	-
18	↕	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
19	‼	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
20	¶	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
21	§	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
22	▬	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
23	↨	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
24	↑	55	7	87	W	119	t	151	ù	183	À	215	Î	247	¸
25	↓	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
26	→	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
27	←	58	:	90	Z	122	z	154	Ü						
28	∟	59	;	91	[	123	{	155	ø						
29	↔	60	<	92	\	124	|	156	£						
30	▲	61	=	93	]	125	}	157	Ø						
31	▼	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
		63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	space

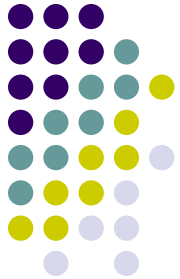
Existují různé regionální varianty této druhé půlky tabulky
(tato varianta neobsahuje české znaky)

Unicode



- Univerzální kód
- Měl by umět zakódovat všechny znaky, které se používají na celém světě (tedy i čínské, korejské,arabské, japonské písmo atd.)
- Existuje více variant tohoto kódu
- Protože množství používaných znaků je velké, používá unicode pro zakódování jednoho znaku **dva bajty - 16 bitů**
- Je-li použito 16 bitů, potom lze používat 65536 různých znaků (2^{16})
- Vzkaz „AHOJ“ by v tomto kódu byl uložen pomocí 8 bajtů (to je dvakrát více než by použil ASCII kód)

Strojový kód



- Binární kód

Číslo → bity

- ASCII kód, Unicode

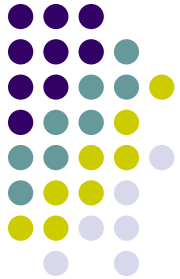
Znak, písmeno, text → bity

- Strojový kód

Program, instrukce, povely → bity

Počítače a mikroprocesory zpracovávají data podle zadaného programu. Program i data jsou zakódována v digitální podobě pomocí kombinací bitů. Strojový kód slouží k zakódování programu. Každá instrukce je zakódována určitou kombinací bitů.

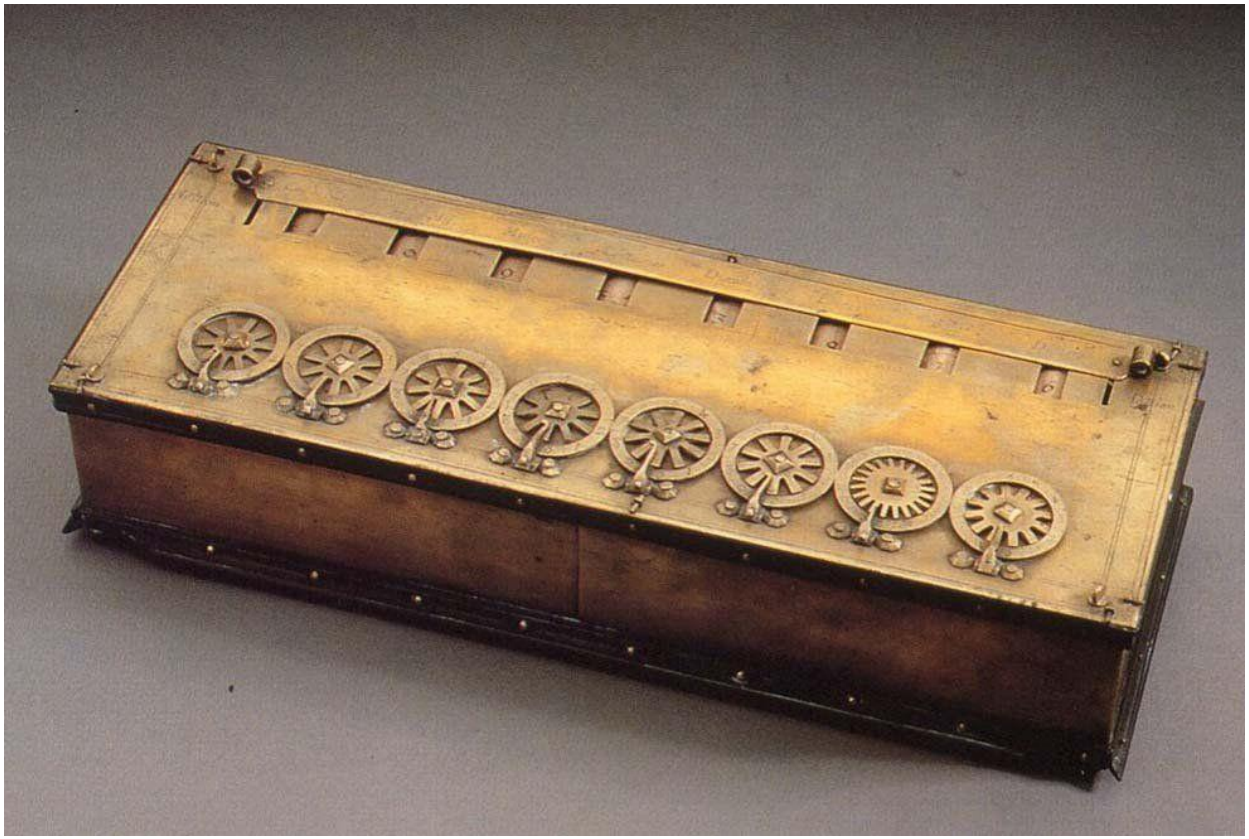
První „Počítače“



- Různé mechanické pomůcky pro usnadnění výpočtů vznikají již v dávnověku a středověku
- Nejedná se však o stroje na zpracování obecných dat – umí pouze provádět nějaký konkrétní typ výpočtu s čísly
- Tyto „počítače“ nelze programovat

První „Počítače“

- Pascalina (1642) - Blaise Pascal sestrojil mechanickou kalkulačku z ozubených koleček pro sčítání a odčítání



První „Počítače“

- Kroková kalkulačka – **G. W. Leibnitz** sestrojil mechanickou kalkulačku založenou na dvojkové soustavě - uměla dokonce násobit, dělit a počítat druhou mocninu

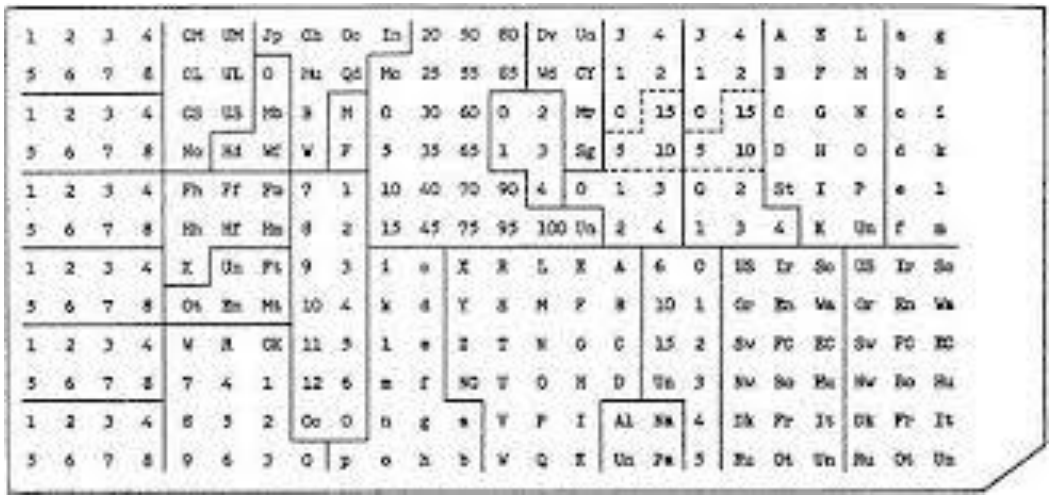
První „Počítače“

- V roce 1820 se začínají sériově vyrábět první kalkulačky – stroj nazvaný Arithmometer

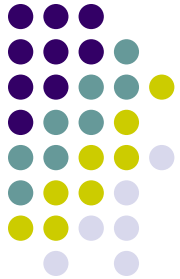
První programovatelné stroje

- programovatelnost byla velmi omezená úrovni
- V podstatě šlo o možnost práce s variabilními daty uloženými na nějakém médiu, které se podobalo pozdějším děrným štítkům

- Za takový stroj lze považovat i různé hudební „hrací skříňky“
- Joseph Marie Jaquard sestrojil tkalcovský stav se tkaným vzorem uloženým na děrném štítku
- Herman Hollerith - stroj na zpracování údajů při sčítání obyvatelstva (děrované kartičky)



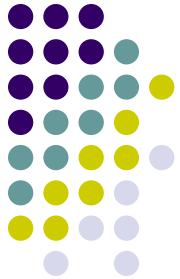
První univerzální počítače



- **Charles Babbage**, britský matematik, byl jedním z prvních geniálních vynálezců v oblasti výpočetní techniky
- Za svého života stihl sestrojit a zprovoznit několik strojů poháněných párou
- Bohužel návrhy jeho nejzajímavějších strojů zůstaly jen na papíře, nemohl je sestrojit z finančních důvodů, ale také proto, že vyžadovaly velmi jemnou mechanickou práci, která by v té době byla obtížně realizovatelná

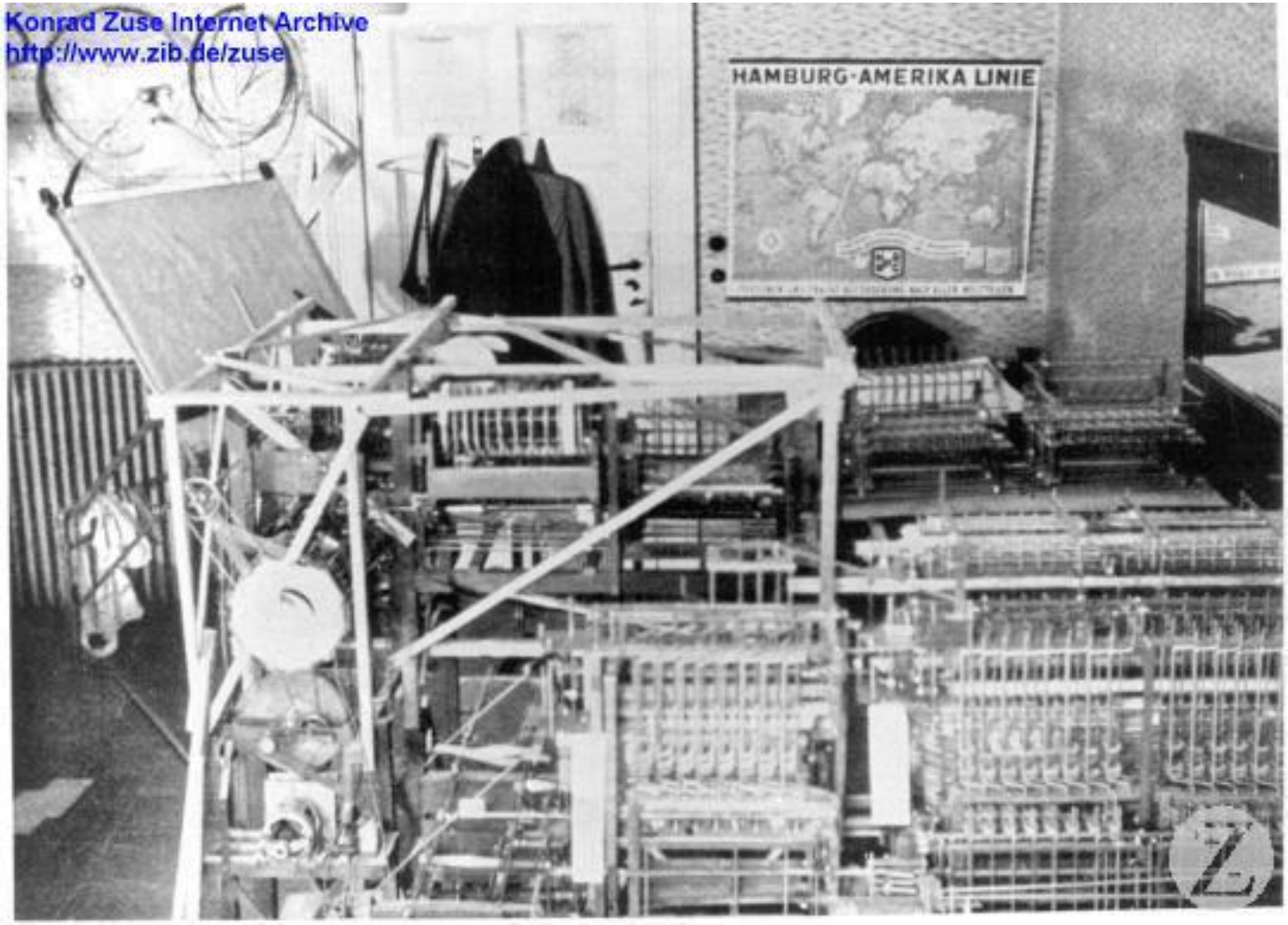
- **Analytický stroj** (Analytical Machine) je považován za návrh prvního univerzálního počítače (rok 1837)
- **Univerzální** znamená, že dokáže simulovat jakýkoliv jiný stroj, algoritmus nebo výpočet bez nutnosti hardwarové přestavby, jen s úpravou programu)
- Jednalo se o programem řízený mechanický číslicový počítač poháněný parou
- Uměl provádět veškeré aritmetické výpočty s operandy, které se ukládaly do dvou registrů
- Dále uměl cykly a větvení programu dle podmínky
- Vstupní data a program se ukládaly na děrný štítek
- **Stroj nikdy nebyl sestaven**, ala přesto pro něj byly napsány první programy na světě
- Hraběnka Ada Lovelace napsala **první program** – výpočet Bernoulliho čísel (později byl po ní pojmenován programovací jazyk ADA)

První číslicový počítač

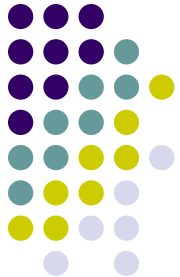


- První číslicové počítače vznikly v nacistickém Německu (což málokdo ví a USA to neradi přiznávají) a projekty byly odtajněny až v 70.letech
- **Konrad Zuse** – německý vědec, první prototyp počítače sestavil již ve svých 28 letech
- **Z1** (1936)
  - paměť 1408 bitů složená z mechanických posuvných prvků
  - Jedinou elektrickou součástkou byl motor
  - Počítač prováděl jednu pracovní fázi za jednu sekundu – pracoval s taktovací frekvencí 1 Hz
  - 22 bitová aritmetika (22-bitově zakódovaná reálná čísla)
  - Instrukce násobení velmi pomalá (řešená postupným opakovaným sčítáním)

Z1


Konrad Zuse Internet Archive
http://www.zib.de/zuse
HAMBURG-AMERIKA LINIE

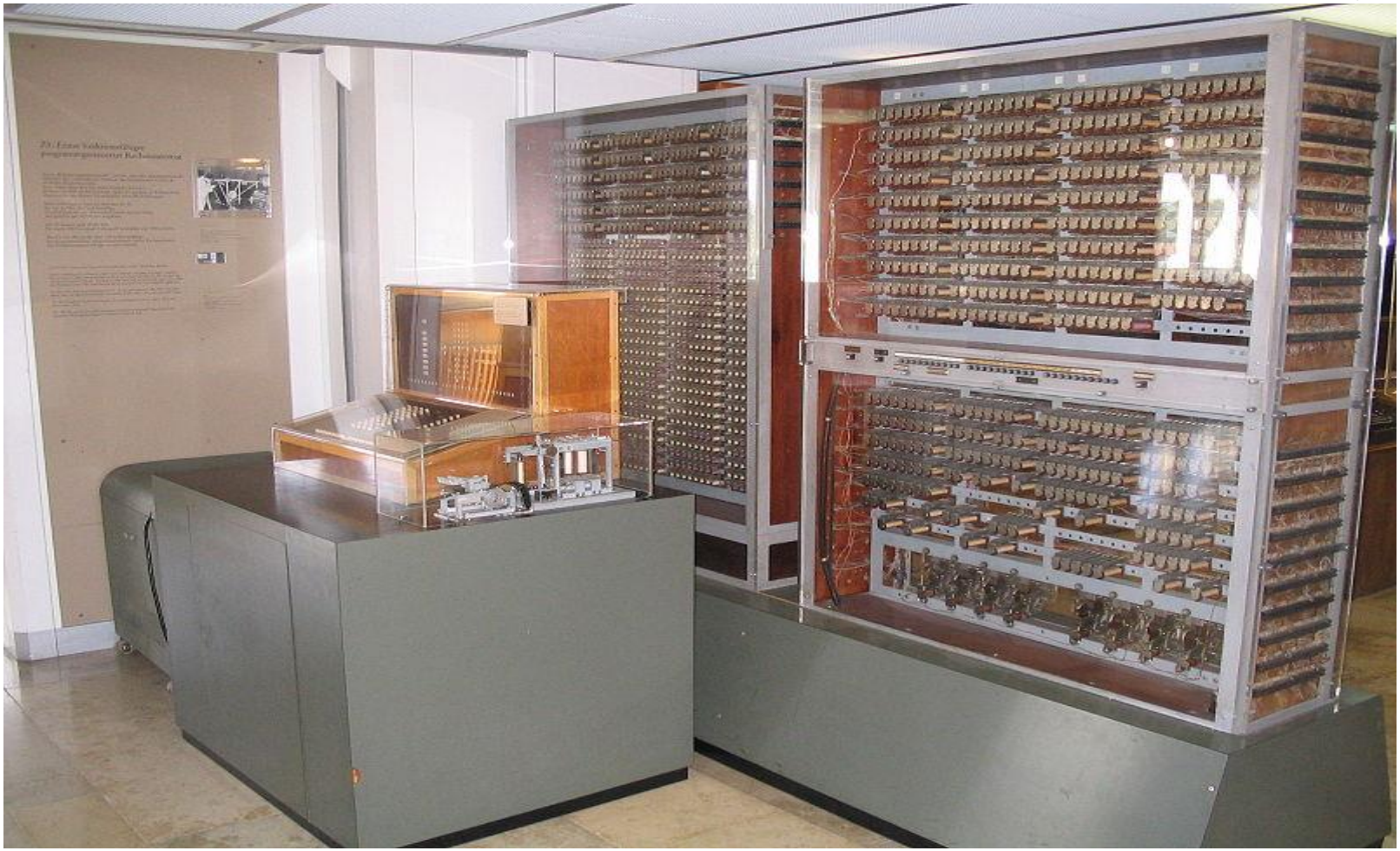
Z2 a Z3



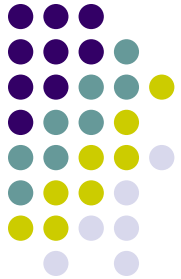
- **Z2** (1939)
  - Pro aritmetickou jednotku použita **elektromagnetická relé** (600 relé)
  - Pracoval pouze s celými čísly s šířkou **16 bitů**
  - Paměť (64x16 bitů) a řadič dále mechanický
  - Rychlost 5 taktů za sekundu. Sčítání trvalo 0,8 sekundy (4 takty)
  - Hmotnost 300 kg. Příkon 1000 W.

- **Z3** (1941)
  - 22-bitový počítač (s reálnými čísly)
  - používaný pro výpočet drah balistických raket V2
  - Uměl sčítat, odčítat, násobit a dělit, počítat odmocninu apod., ale neměl podmíněný skok (program nelze rozvětvit)
  - Celkem 2600 relé, hmotnost 1000 kg
  - násobení trvalo 3 až 5 sekund

Z3

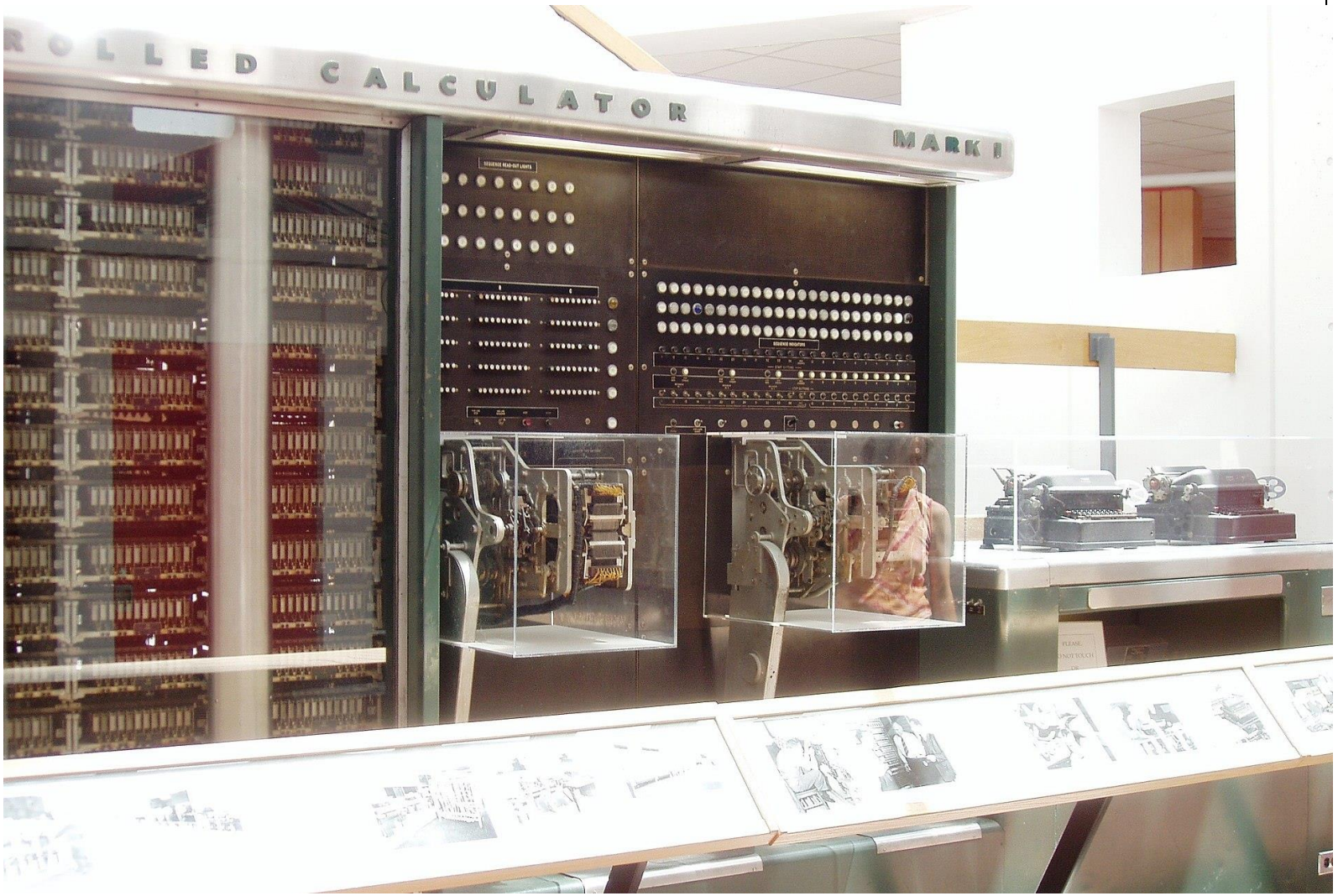


Mark I

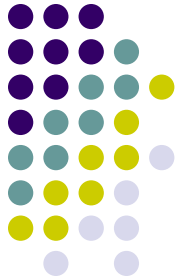


- V letech 1937 - 1943 se v USA v laboratořích firmy **IBM** pod vedením Američana **Howarda Aikena** konstruoval elektromechanický počítač **Mark1**
- Základním stavebním prvkem počítače Mark1 byly **reléové obvody** (obsahuje jich 3500)
- V podstatě šlo o první plně elektronický fungující počítač, který (dle americké verze dějin) zahájil éru tzv. nulté generace počítačů
- V roce 1944 na něm byly prováděny výpočty simulací americké atomové bomby
- Hmotnost 35 tun

Mark 1

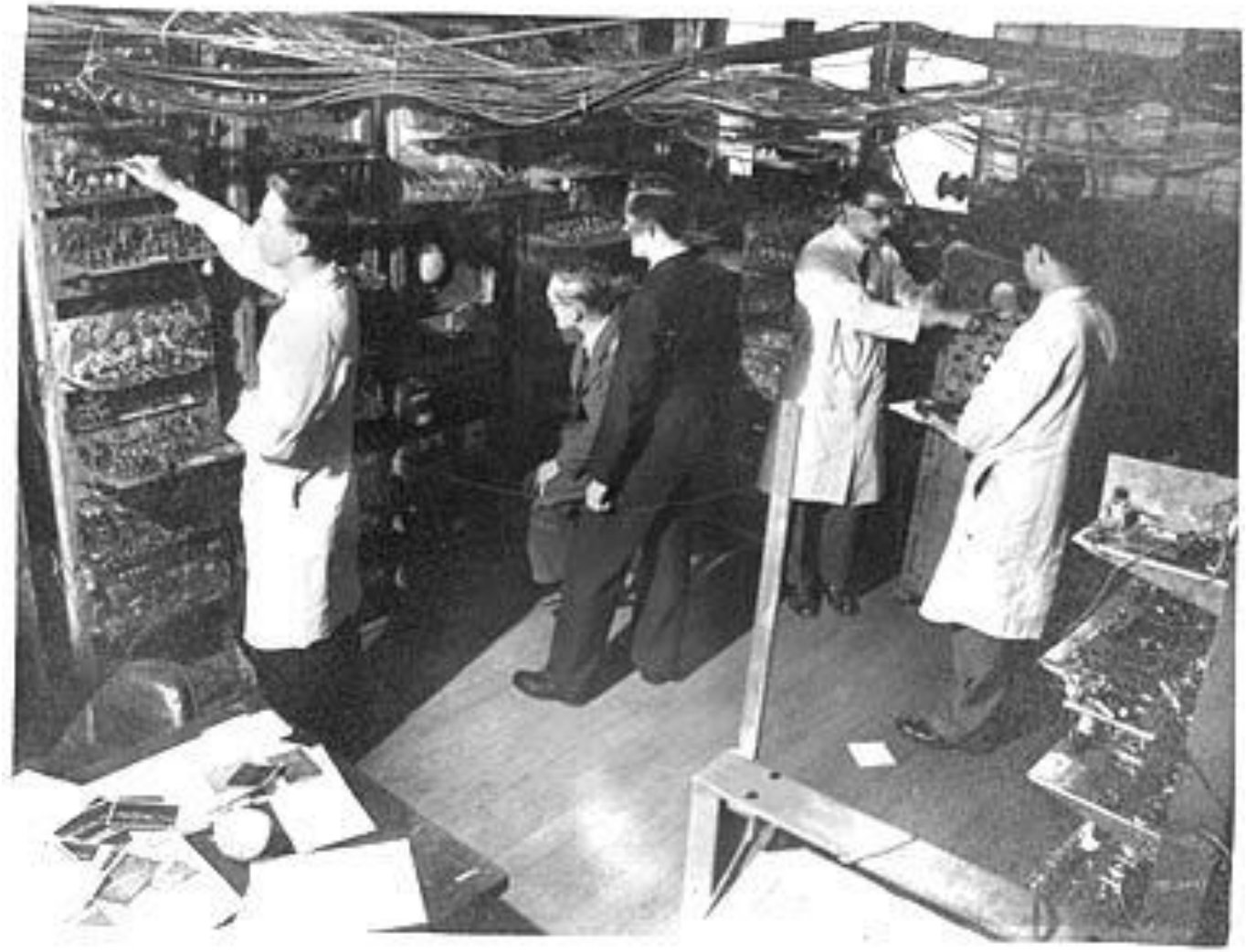


EDSAC

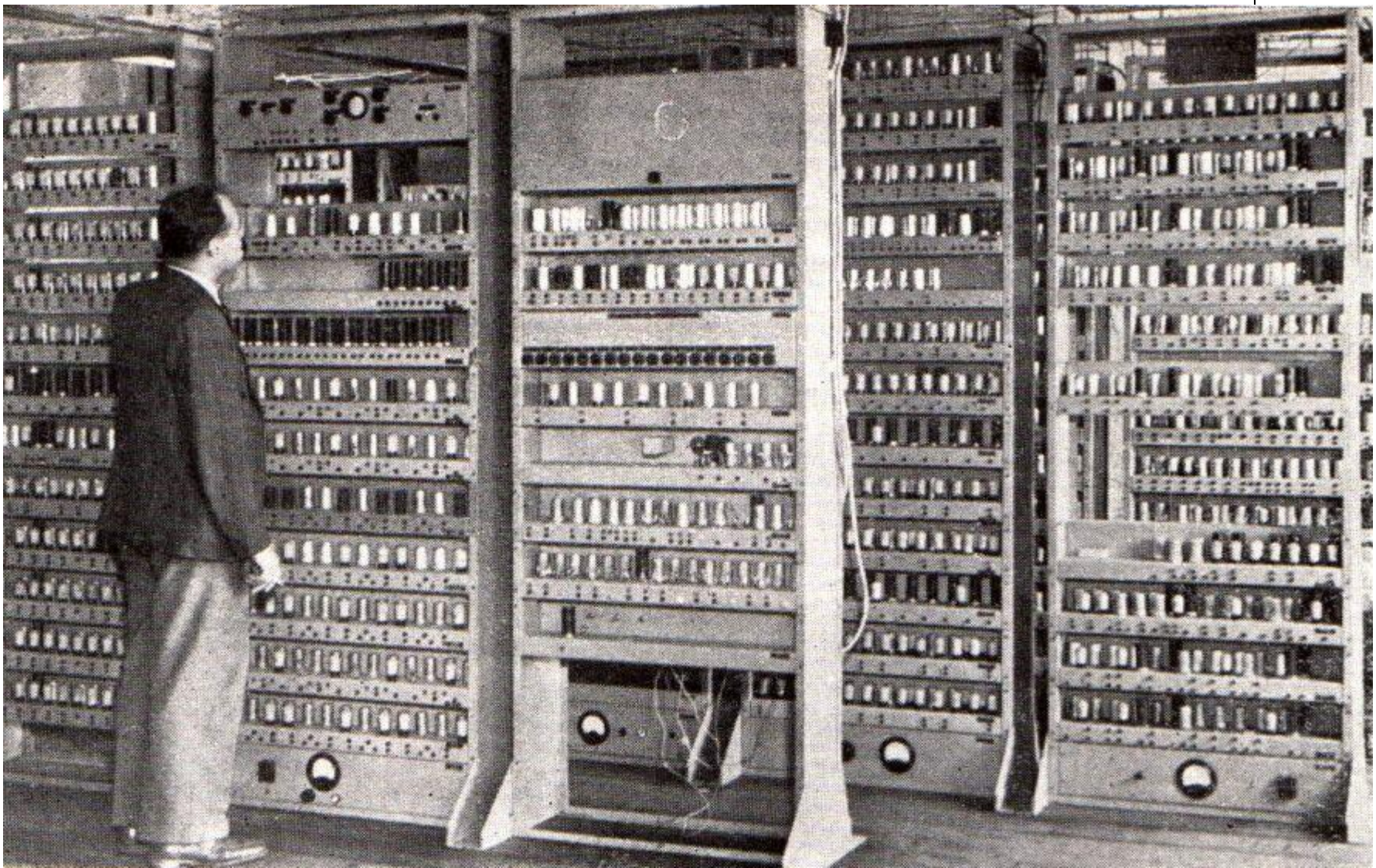


- 1948 - Electronic delay storage automatic calculator (EDSAC)
- První britský číslicový počítač
- Stroj navrhl **Maurice Wilkes**
- Jednoduché operace zvládal za 1,5 ms, násobení za 6 ms
- spotřeba 11 000 W

EDSAC (během vývoje)



EDSAC

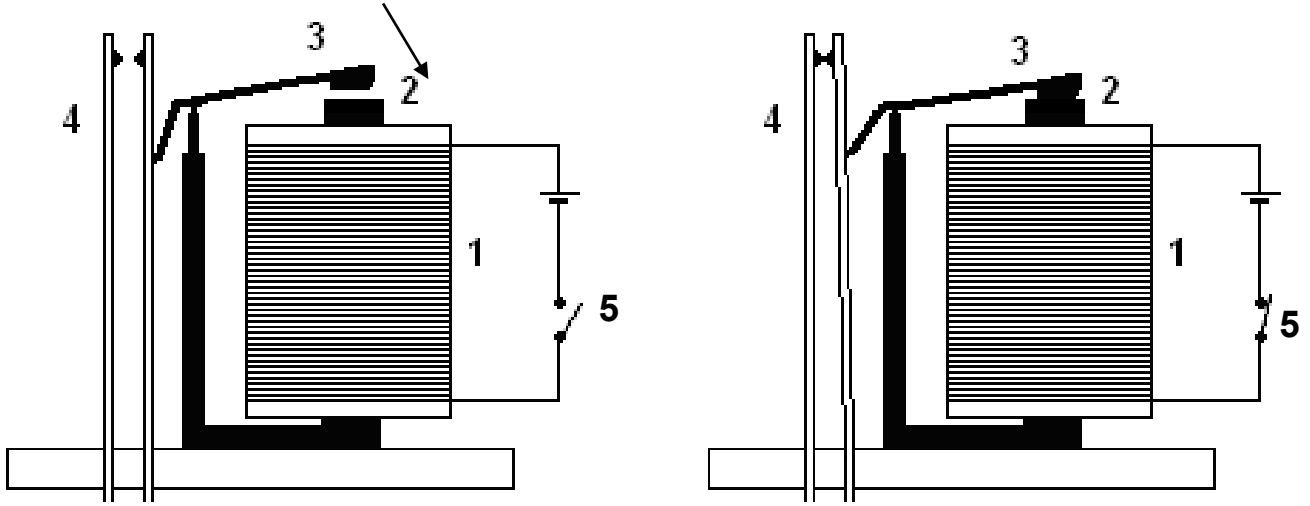


Nultá generace počítačů

- Počítače se z historického hlediska klasifikují podle toho, jaké základní stavební prvky obsahují
- Podle toho se počítače dělily do **generací**.
- První počítače (někdy označované jako nultá generace) byly založeny na mechanických nebo elektromechanických prvcích, zejména zde dominovala **relé**

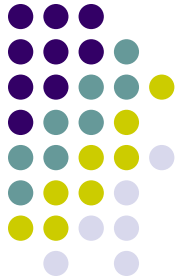
Elektromagnetické relé

- Bylo vynalezeno v roce 1835 Josefem Henrym
- Skládá se z **elektromagnetu** a **pohyblivého kontaktu**, který podpírá tzv. **kotva**.
- Po přivedení **proudu** do **cívky elektromagnetu** vznikne přitažlivá síla, která pohne kotvou a ta **přepne** pohyblivý kontakt.
- Po **odpojení** proudu se kotva a kontakt vrátí do **klidového stavu**.
- Obvykle bývá ovládací spínací výkon, potřebný k přitažení kotvy, mnohem menší než ovládaný spínaný výkon



1-elektromagnet, 2-jádro elektromagnetu, 3-kotva, kterou elektromagnet přitáhne, 4-pružné kontakty spínaného obvodu, 5-spínací obvod

Elektromagnetické relé



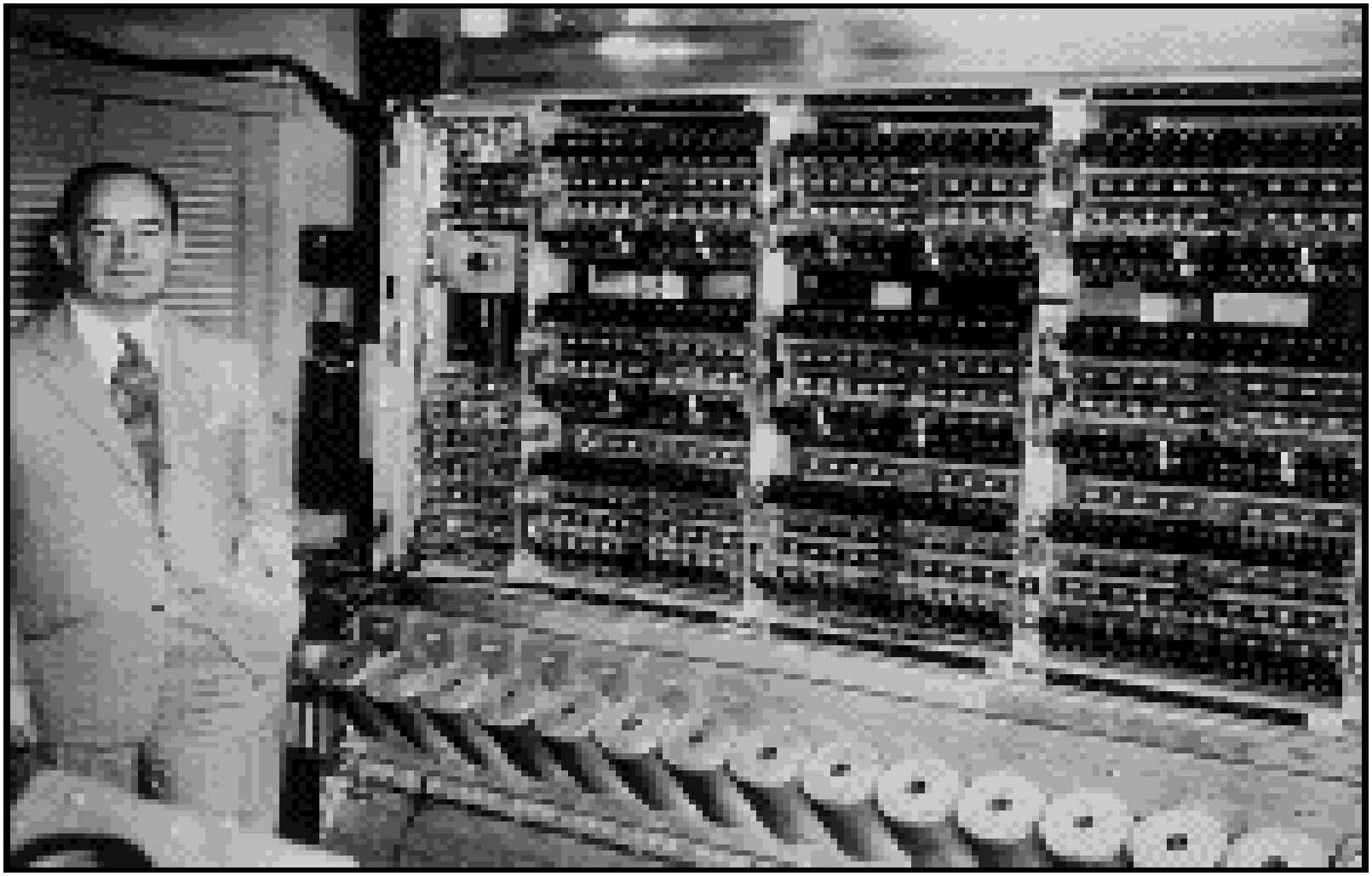
- Výhodou je **galvanické oddělení** spínacího a spínaného obvodu

- Nevýhody
  - Vysoká hmotnost
  - Velké rozměry
  - Vysoká cena (cívka, jádro…)
  - Je **pomalé** – velmi nízká spínací rychlost značně omezovala výpočetní výkon prvních počítačů
  - Při spínání a rozepínání mohou vznikat zákmity, jiskření

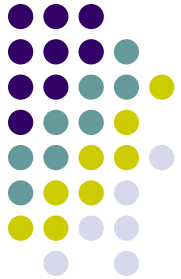
John Von Neumann

- Jeden z nejšpičkovějších matematiků své doby
- Jsme mu vděčni za vynalezení architektury počítače v podobě, v jaké je dosud známe
- Von Neumann neměl elektrotechnické vzdělání a neuměl počítač postavit, byl pouze autorem myšlenky, jak by měl takový počítač vypadat a pracovat
- Narodil se v roce 1903 v Budapešti
- V roce 1928 vymyslel nové odvětví matematiky – tzv. teorii her, která se zabývá výpočty strategií, jak hrát tu či onu hru nebo jak se aspoň vyhnout prohrám
- K ověřování svých výpočtů a algoritmů potřeboval stroj, který prováděl automaticky zadané operace
- ... a tak se zrodila idea programovatelného počítače
- Zemřel mlád v roce 1957.... tedy ještě před vynálezem integrovaných obvodů, vyrobením prvního mikroprocesoru a zrodem moderní výpočetní techniky

John Von Neumann



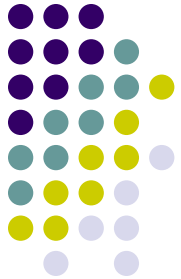
Charakteristické vlastnosti Von Neumannova počítače



- Teorie koncepce jeho počítače byla prezentována na vědecké konferenci v roce 1945
- do doby před Von Neumannem byly počítače pouze jednoúčelové stroje, které byly vyrobeny za účelem jedné řešení konkrétní úlohy (pro jinou úlohu se musely přestavět)
- Hlavní myšlenka: Struktura je nezávislá od zpracovávaných problémů, na řešení problému se musí zvenčí zavést návod na zpracování, program a musí se uložit do paměti, bez tohoto programu není stroj schopen práce
- Program a zpracovávaná data jsou uloženy v paměti společně

Veškeré přizpůsobení konkrétním úkolům má být řešeno výhradně programem

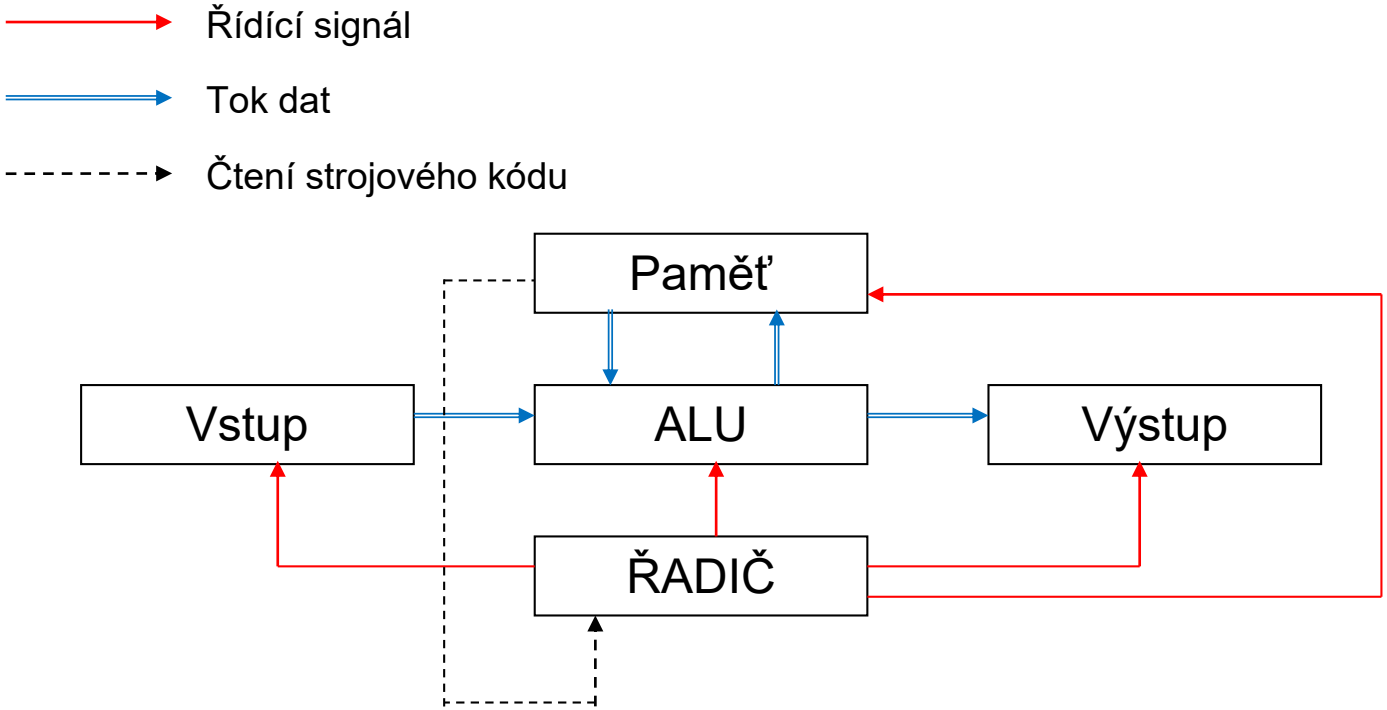
Charakteristické vlastnosti Von Neumannova počítače



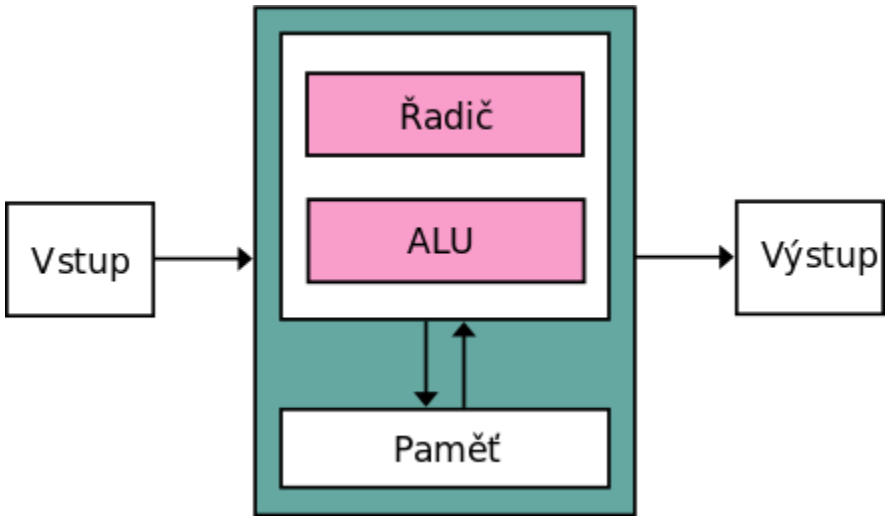
- Využití **dvojkové soustavy** - Všechna data (instrukce, adresy,…) jsou binárně kódované, správné dekódování zabezpečují vhodné logické obvody v řídící jednotce
- **5 funkčních jednotek** – řídící jednotka, aritmeticko-logická jednotka, paměť, vstupní zařízení, výstupní zařízení
- **Paměť** je rozdělená na stejně velké buňky, které jsou průběžně očíslované, přes číslo buňky (**adresu**) se dá přečíst nebo změnit obsah buňky
- Po sobě jdoucí instrukce programu se uloží do paměťových buněk jdoucích po sobě, přístup k následující instrukci se uskuteční z řídící jednotky zvýšením instrukční adresy o 1
- Instrukcemi skoku se dá odklonit od zpracování instrukcí v uloženém pořadí

- Neuvěřitelné, že i dnes po mnoha letech prakticky beze změn vše výše uvedené platí i pro dnešní běžné počítače

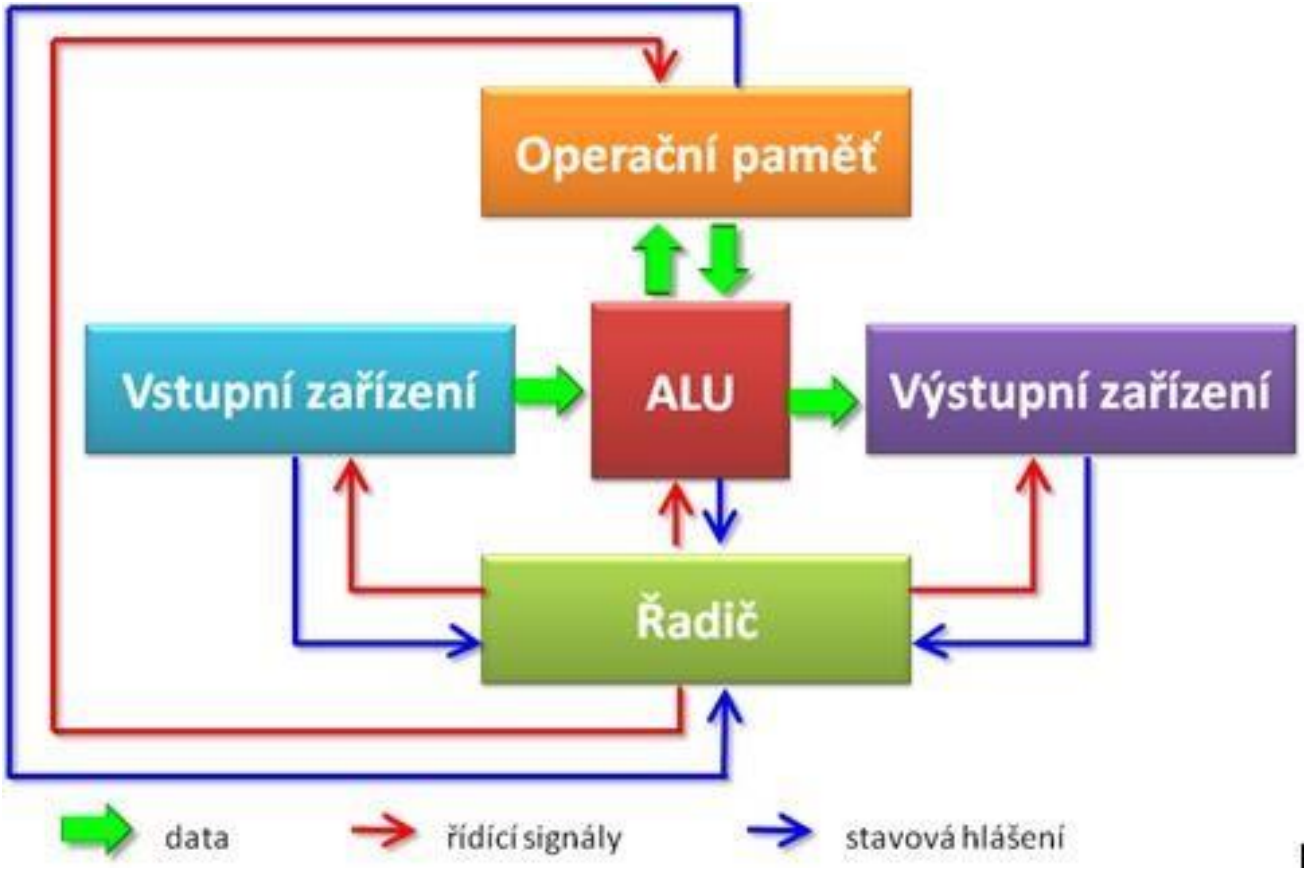
Von Neumannova architektura



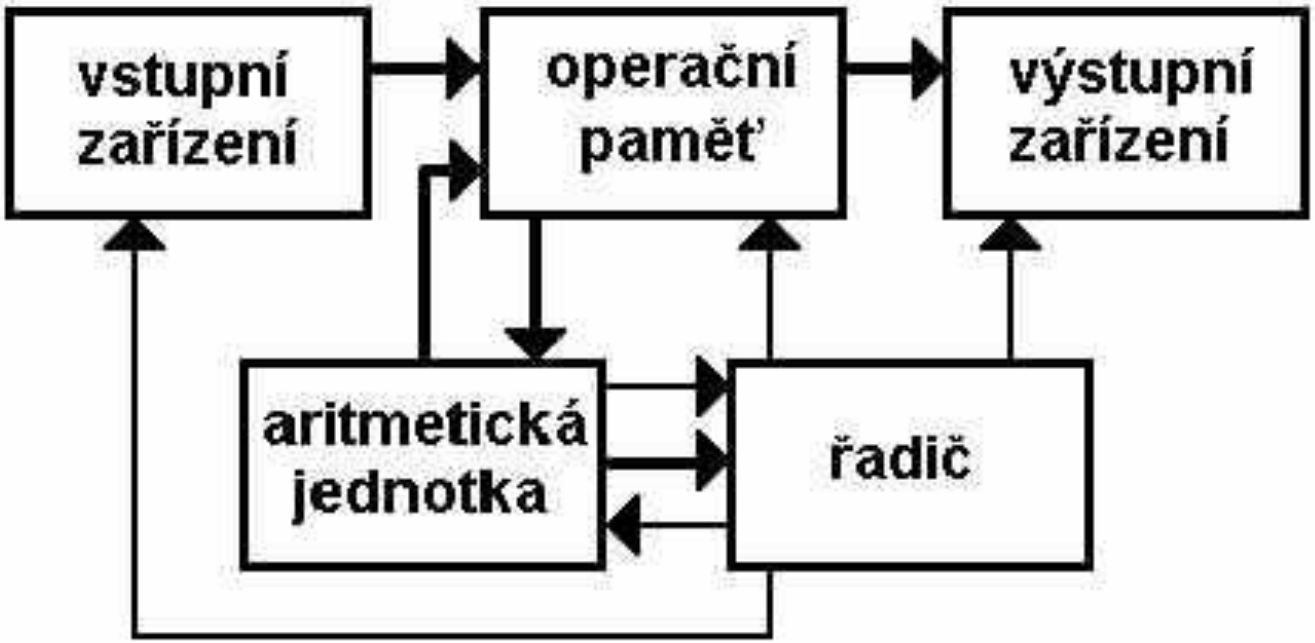
Von Neumannova architektura



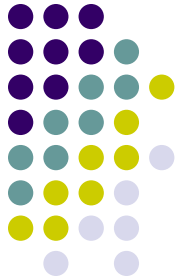
Von Neumannova architektura



Von Neumannova architektura

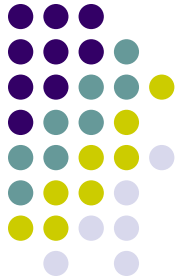


Řadič



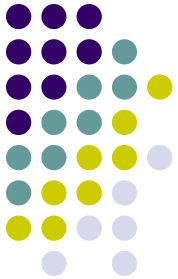
- Srdcem Von Neumannova počítače je řadič, který svým způsobem řídí chod celého počítače.
- Řadič = **řídící jednotka**
- Anglicky se tato jednotka nazývá **Controller**
- Řadič **generuje řídící signály** pro ostatní bloky počítače – dává ostatním jednotkám povely k činnosti (např. ALU posílá signál, jakou aritmetickou operaci má provést, paměti potom povel k zápisu výsledku na vybranou adresu atd...)
- Řadič rozumí **strojovému kódu**, kterým jsou zakódované povely tvořící program. Bajty strojového kódu jsou řadičem vybírány z paměti, dekódovány a pak dochází vykonání instrukce
- Provedení každé instrukce se vždy skládá z určitého počtu dílčích kroků (fází), které mohou provádět různé části počítače samy, či ve vzájemné spolupráci, vždy ale za potřebné vzájemné součinnosti.
- Tuto koordinaci zajišťuje právě řadič. Přitom vychází z právě prováděné strojové instrukce, od které pak odvozuje svůj postup (postupně vysílá ostatním jednotkám řídící signály vedoucí k provedení instrukce)

ALU



- Aritmeticko-logická jednotka
- „dělník“ vykonávající operace s daty
- Obsahuje binární sčítačku, násobičku, komparátory, příznakové bity….
- S využitím těchto jednotek dokáže provádět veškeré operace s bajty
- Řadič dává ALU povel, jakou operaci má vykonat s daty na jejím vstupu. Před tím řadič zařídil, aby byl z paměti např. přečten bajt se kterým se bude pracovat.

Paměť



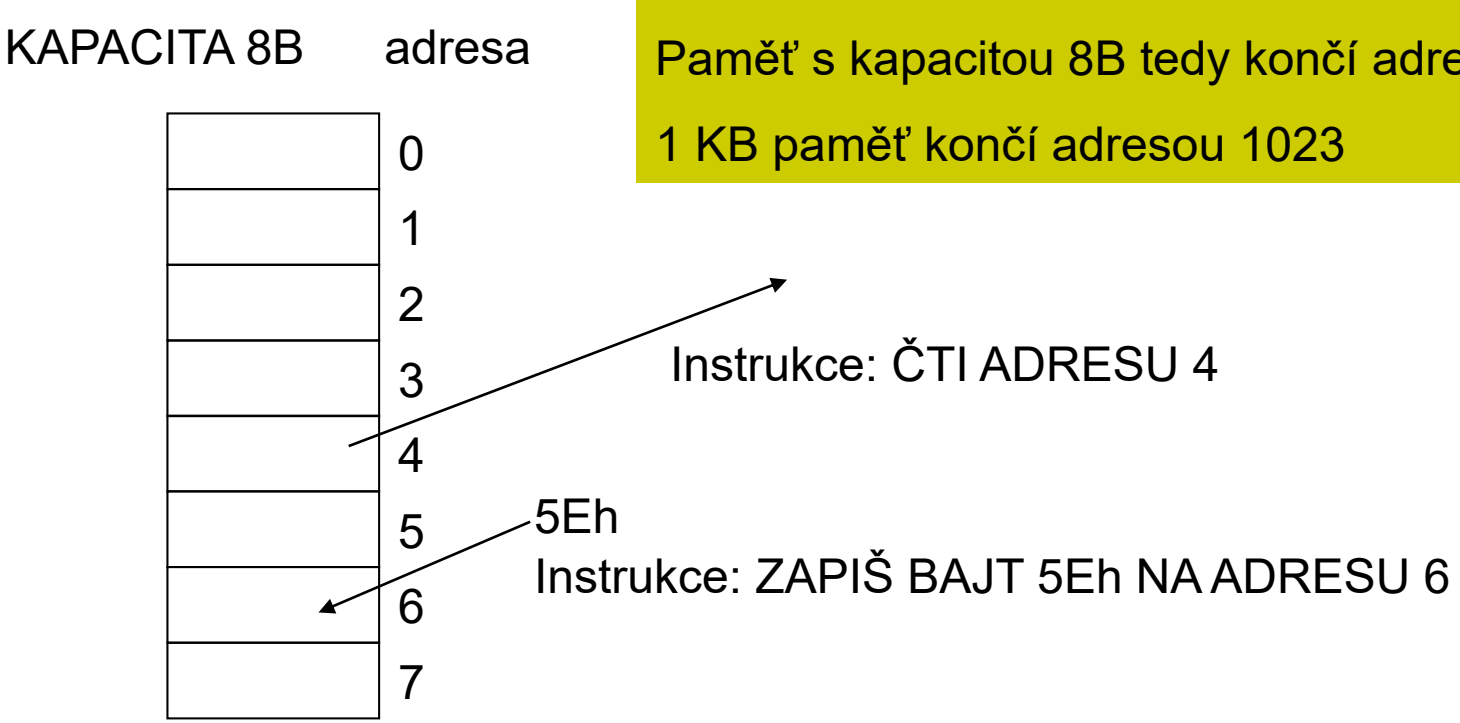
- Obecně je paměť cokoliv, kam lze **uložit data** a později je opět vyzvednout
- Paměť je tedy **úložiště informací** (bitů, bajtů)
- O různých typech pamětí se naučíme později
- Jednou ze základních myšlenek Von Neumannovy architektury je **adresovatelná paměť**
- Paměť si lze představit jako mnoho „přihrádek“ sloužících k uložení jednoho bajtu (data obvykle ukládáme po bajtech)
- Všechny takové přihrádky jsou postupně očíslovány, aby byla každá z nich jednoznačně identifikovatelná
- Každá paměťová „přihrádka“ tedy má svou **adresu**
- Při práci s pamětí je třeba vždy vybrat **konkrétní adresu**, na kterou se bude zapisovat bajt nebo ze které se bude číst

Adresovatelná paměť

Adresace začíná vždy od nuly.

Paměť s kapacitou 8B tedy končí adresou 7

1 KB paměť končí adresou 1023



Instrukce pracují vždy s konkrétním paměťovým místem, které je určeno adresou

Výpis paměti

První bajt leží v paměti na adrese 0 a má hodnotu 27h

Bajt s hodnotou 52h leží na adrese 1

Na adrese 10 leží bajt 5Dh

Tento bajt leží na adrese 7

Na adrese 15 leží bajt s hodnotou 77h

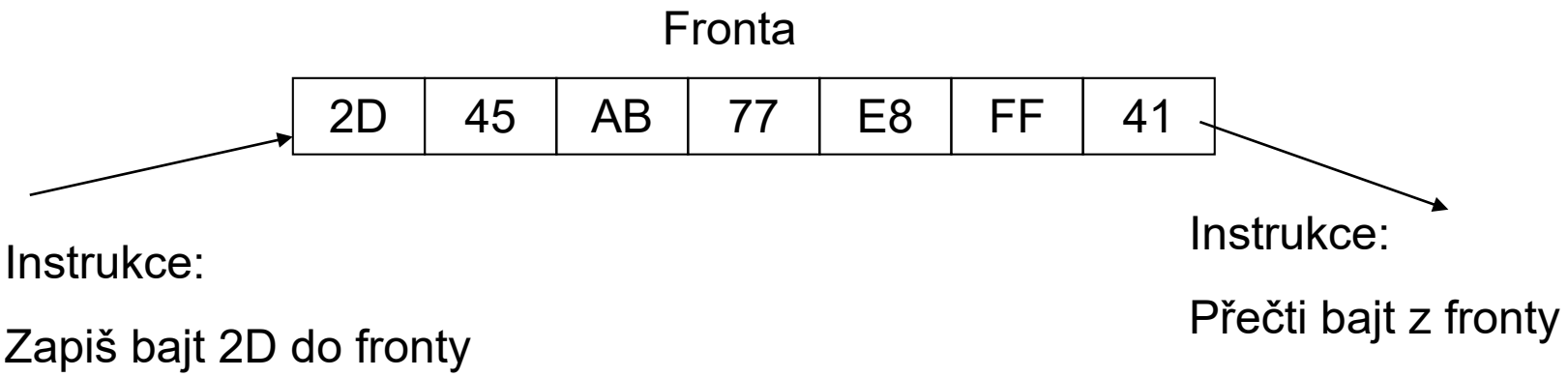
adresa								
0000	27	52	B4	FF	00	5C	AA	5E
0008	F1	7C	5D	99	01	02	E4	77
0010	11	22	F0	C2	C2	4D	E7	00
0018	00	00	05	FF	00	FF	00	7F
0020	51	44	32	41	A0	B2	CD	73
0028	76							

Výpis obsahu paměti se zobrazuje zásadně hexadecimálně (je to prostorově nejúspornější)

Další úspora se dosahuje výpisem po řádcích - na jednom řádku je několik bajtů vedle sebe a na začátku řádku je uvedena pouze adresa prvního z bajtů, další bajty leží na následujících adresách

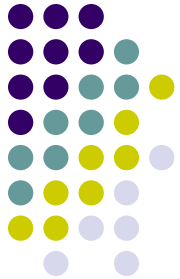
Paměti bez adresace
Fronta

- Kromě adresovatelné paměti existují i další možné paměťové modely
- Fronta je typem paměti bez adresace, do které vkládáme bajty a při čtení je vybíráme ve stejném pořadí, v jakém byly vloženy
- Fronta = **FIFO** (First In First Out)



Instrukce nepracují s konkrétním paměťovým místem. Nefunguje zde žádná adresace

Paměti bez adresace
Zásobník

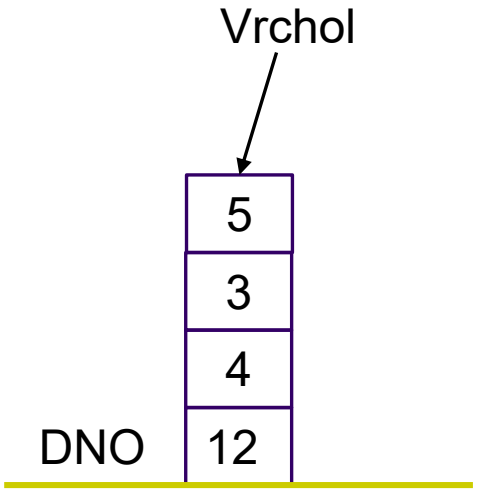


- Zásobník je také typem paměti bez adresace
- Lze si ho představit jako odkládání bajtů **na hromadu**
- Bajty se ukládají na **vrchol zásobníku**
- Při čtení se bajty odebírají z vrcholu
- **Naposledy uložený** bajt tedy bude přečten jako **první**
- A naopak první uložený bajt bude přečten až nakonec, po odebrání všech bajtů ležících „v hromadě“ nad ním
- Zásobník = **LIFO** (Last In First Out)

Zásobníková paměť

- Výpočet: (5+3)x4 – 12

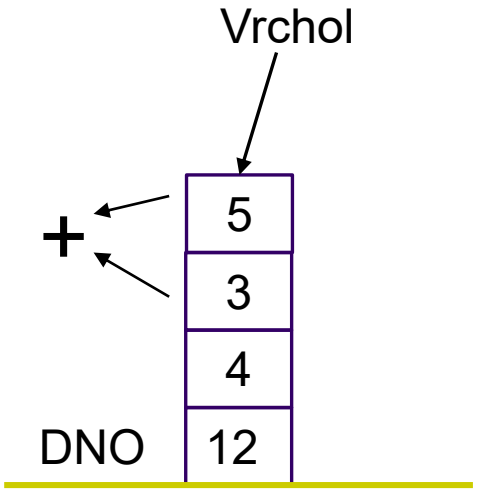
- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5



Zásobníková paměť

- Výpočet: (5+3)x4 – 12

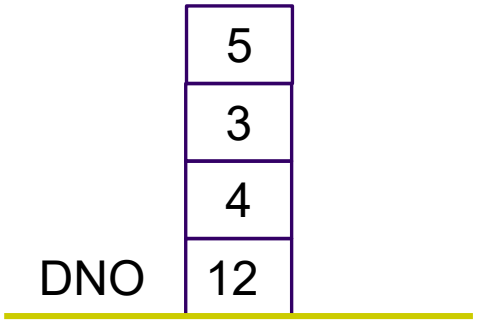
- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5
- Sečti



Zásobníková paměť

- Výpočet: (5+3)x4 – 12

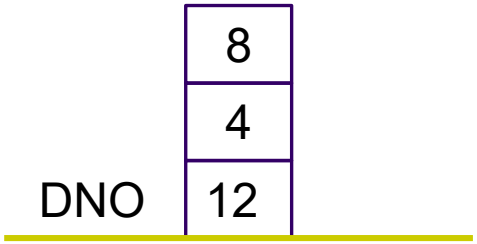
- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5
- Sečti



Zásobníková paměť

- Výpočet: (5+3)x4 – 12

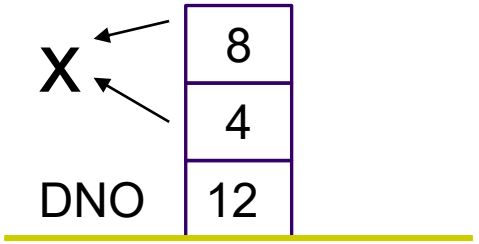
- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5
- Sečti



Zásobníková paměť

- Výpočet: (5+3)x4 – 12

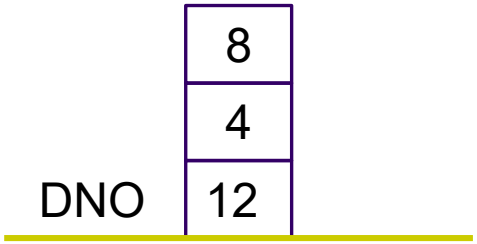
- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5
- Sečti
- Vynásob



Zásobníková paměť

- Výpočet: (5+3)x4 – 12

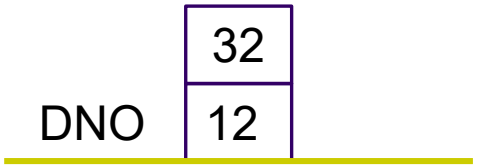
- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5
- Sečti
- Vynásob



Zásobníková paměť

- Výpočet: (5+3)x4 – 12

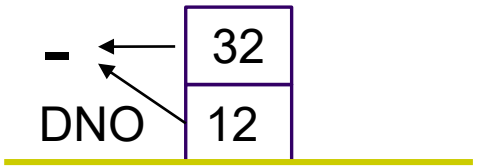
- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5
- Sečti
- Vynásob



Zásobníková paměť

- Výpočet: (5+3)x4 – 12

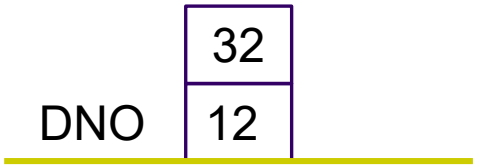
- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5
- Sečti
- Vynásob
- Odečti



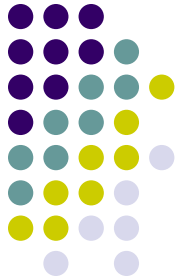
Zásobníková paměť

- Výpočet: (5+3)x4 – 12

- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5
- Sečti
- Vynásob
- Odečti



Zásobníková paměť

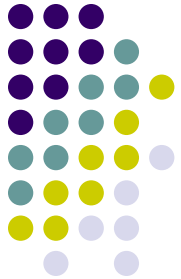


- Výpočet: (5+3)x4 – 12

- Ulož do zásobníku 12
- Ulož do zásobníku 4
- Ulož do zásobníku 3
- Ulož do zásobníku 5
- Sečti
- Vynásob
- Odečti

DNO | 20

K výpočtu se vždy odebírají bajty z vrcholu zásobníku. Výsledek se vždy ukládá na vrchol zásobníku Při přístupech do paměti (ukládání / čtení dat) se neudává, se kterým paměťovým místem chceme pracovat.



- Počítače Von Neumannovy architektury používají pro uložení programu a dat **zásadně adresovatelnou paměť**

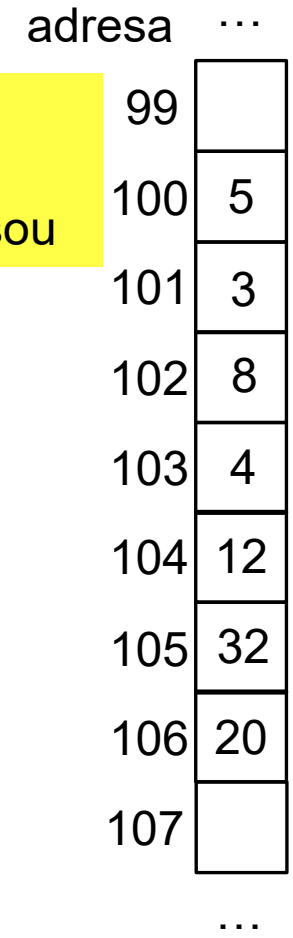
- Zásobník a fronta mohou být také použity, ovšem pouze pro různé vedlejší účely (nikdy ne jako hlavní operační pamět)
- Paměť se zásobníkovým modelem používají tzv. zásobníkové architektury (PostScript, PicoJava…)

Adresovatelná paměť

- Výpočet: (5+3)x4 – 12

Při každém přístupu do paměti se pracuju s konkrétní vybranou adresou

- Ulož 5 na adresu 100
- Ulož 3 na adresu 101
- Ulož 4 na adresu 103
- Ulož 12 na adresu 104
- Sečti bajty z adres 100 a 101, výsledek ulož na adresu 102
- Vynásob bajty z adres 102 a 103, výsledek ulož na adresu 105
- Odečti od bajtu z adresy 105 hodnotu bajtu z adresy 104, výsledek ulož na adresu 106

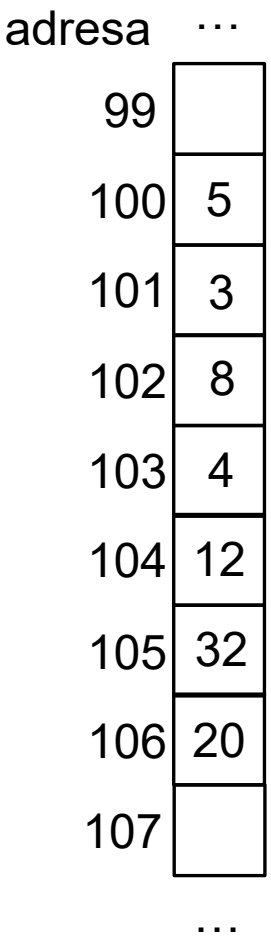


Na Von Neumannově architektuře používáme adresovatelnou paměť a výpočet by probíhal uvedeným způsobem. Vyšší programovací jazyky nám umožňují nadeklarovat si proměnné, do kterých si čísla uložíme – ve skutečnosti je pro každou proměnnou vyhrazeno jedno paměťové místo. Takže například pokaždé, když probíhá práce s proměnnou **a**, pracuje se vlastně s adresou 100

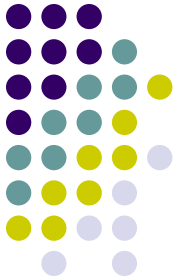
- Výpočet: (5+3)x4 – 12

```
a=5
b=3
c=4
d=12
e=a+b
f=e*c
g=f-d
```

- Ulož 5 na adresu 100
- Ulož 3 na adresu 101
- Ulož 4 na adresu 103
- Ulož 12 na adresu 104
- Sečti bajty z adres 100 a 101, výsledek ulož na adresu 102
- Vynásob bajty z adres 102 a 103, výsledek ulož na adresu 105
- Odečti od bajtu z adresy 105 hodnotu bajtu z adresy 104, výsledek ulož na adresu 106



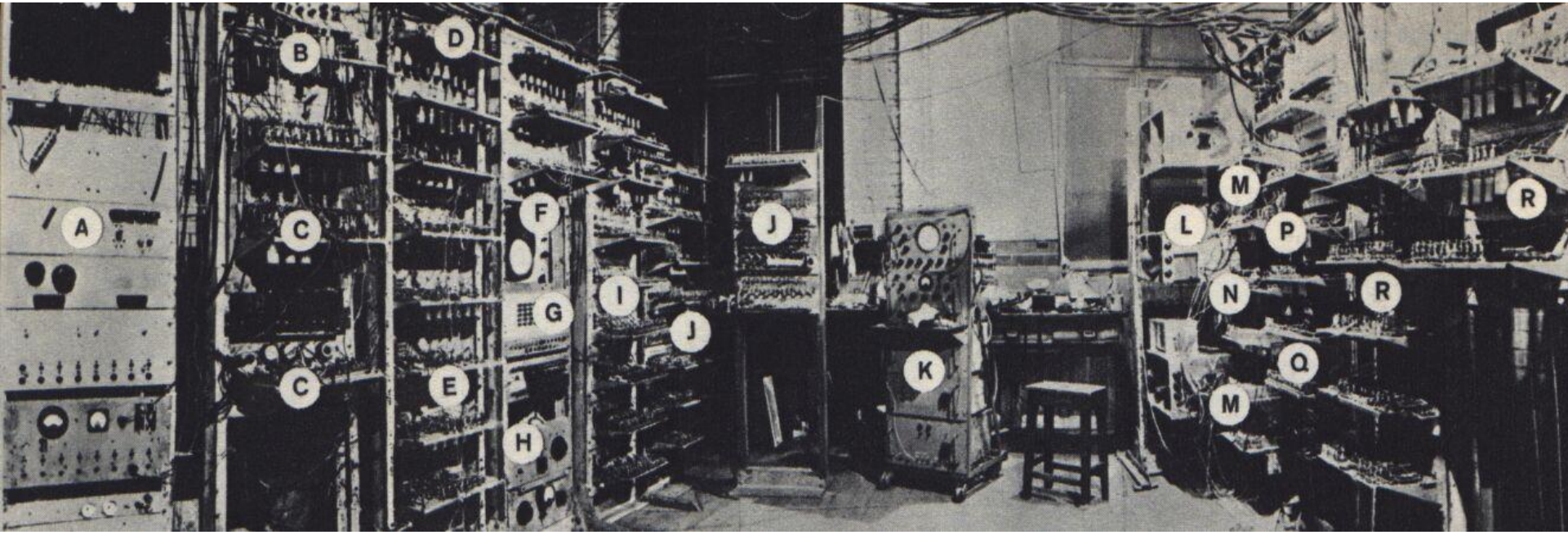
EDVAC



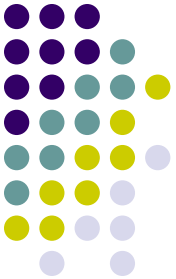
- První počítač von Neumannovy architektury
- První počítač s realizovanou převratnou myšlenkou uložit program do paměti počítače spolu se zpracovávanými daty
- paměťové obrazovky - na stínítku se po osvícení určitého bodu po jistou dobu udrží zbytkový náboj. Ten je možné následně zase přečíst (tak, že se tímto bodem nechá projít další elektronový paprsek, a jeho intenzita se vyhodnotí speciální elektrodou za stínítkem)
- na jedné obrazovce je takto možné uchovávat až 2048 bitů
- Nejprve se programoval ve strojovém kódu
- Později Alan Turing, a vymyslel jednoduchou formu jazyka symbolických adres (assembleru)

EDVAC

A: Napájení zdroje, B: Hlavní oscilátor (100 kHz), C, E: Vychylovací obvody pro obrazovku, D: Obvody výběru adresy, F: Monitor zobrazující obsah paměti, G: Ovládací pult, H: Vysoko napěťový zdroj pro obrazovky, I: Generátor pulsů pro jednotlivá dekadická místa, J: Generátor časovacích signálů, K: Přenosný osciloskop, L: 6 paměťových obrazovek, M: Obvody pro regeneraci paměti, N: Odčítačka, P: Výběrové obvody a hradla, Q: Sčítačka, R: Násobička



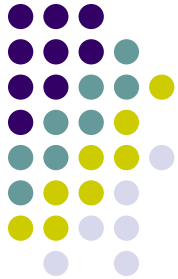
Kde je mikroprocesor ???



- Ve Von Neumanově architektuře nefiguruje žádný blok zvaný **mikroprocesor**
- je dobré si uvědomit, že v době vzniku této architektury žádné mikroprocesory neexistovaly
- počítače v té době byly **stroje ohromných rozměrů** a jednotlivé bloky Von Neumanova schematu byly oddělené „skříně“ obsahující tisíce součástek – např. řadič byl samostatný blok o rozměrech 2x3 metry a stejně tak třeba ALU
- S postupem času, rostoucím pokrokem a stupněm integrace se podařilo zmenšit řadič na jediný plošný spoj, poté na jedinou součástku a nakonec **řadič** a **ALU** splynuly v jedno – vznikl **mikroprocesor**

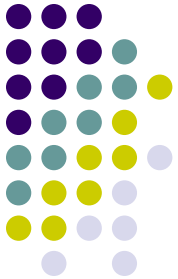
- Další integrací vznikly **jednočipové mikropočítače**, které na jediném čipu obsahují kromě řadiče a ALU i paměť, vstupně výstupní obvody a další bloky (např. časovače, převodníky, sériovou linku...)

Výhody a nevýhody Von Neumannovy architektury



- Von Neumannova architektura je ryze sekvenční a nepředpokládá žádný paralelismus.
- Psát programy pro sekvenční zpracování, je snazší než psaní paralelních programů a i to je zřejmě jeden z důvodů, proč se von Neumannova koncepce udržela až dodnes, zatímco alternativní koncepce, podporující větší míru paralelismu, nejsou zdaleka tak úspěšné
- Další velkou nevýhodou se později ukázala i původní přednost von Neumannovy architektury — snaha o maximální univerzálnost a neměnnou vnitřní strukturu.
- Počítač, který je univerzální, dokáže sice (téměř) všechno, ale občas neefektivně.
- Specializované počítače, jejichž vnitřní struktura je uzpůsobena pro určitý konkrétní typ úloh, dokáží být mnohem efektivnější (například speciální signálové procesory pro zpracování zvuku a obrazu)

Základní odlišnosti dnešních počítačů od von Neumannova schémata



- Počítač může mít více procesorů a provádět více úloh současně
- Počítač může na rozdíl od von Neumannova schématu pracovat nejen pouze v tzv. diskrétním režimu (v průběhu výpočtu již nebylo možné s počítačem dále interaktivně komunikovat)
- V dnešních počítačích existují vstupně-výstupní zařízení (I/O devices), která umožňují jak vstup, tak výstup
- Program se do paměti nemusí zavést celý, ale je možné zavést pouze jeho část a ostatní části zavádět až v případě potřeby.
- Programům lze předstírat „virtuální paměť“
- Jedna instrukce může pracovat s více daty naráz (stejnou operaci provedeme s více čísly)

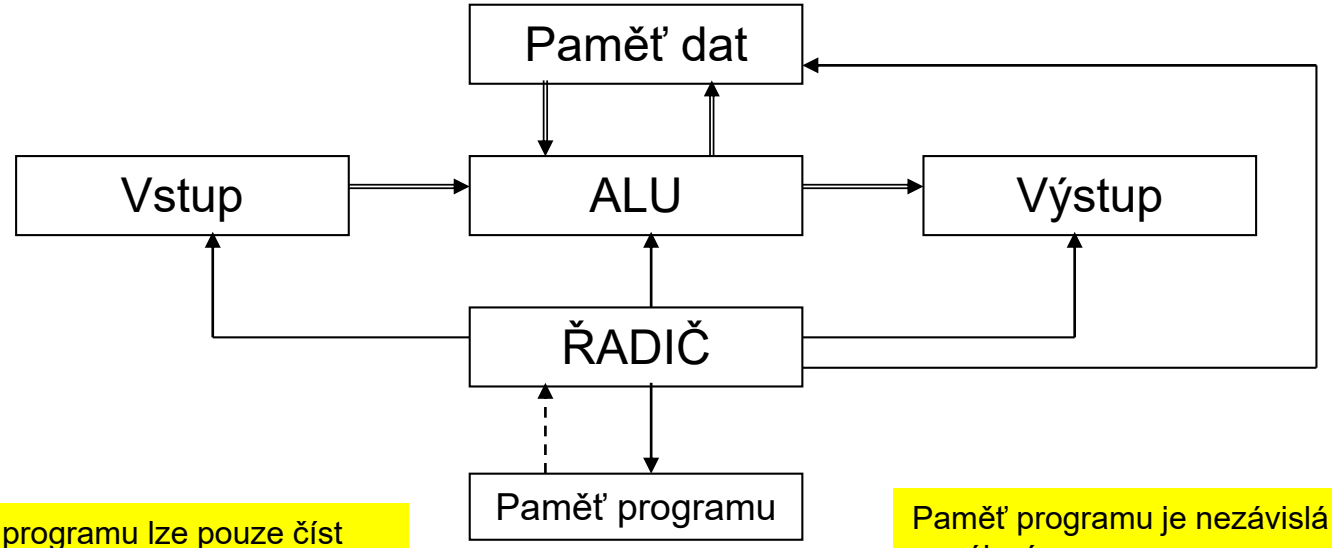
Harvardská architektura

Řídící signál

Tok dat

Čtení strojového kódu

Hlavní rozdíl oproti Von Neumannově architektuře spočívá v oddělené paměti programu a paměti dat

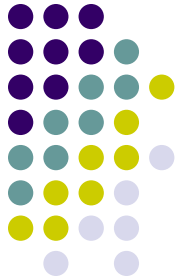


Z paměti programu lze pouze číst strojový kód. Počítač sám neumí do této paměti zapisovat. Program tak nemůže přepsat sám sebe. Počítač nelze „zavirovat¨. Naprogramovat počítač lze pouze zvenku

Paměť programu je nezávislá na napájení

Program zde může být uložen na pevno a je tak k dispozici hned po zapnutí počítače (nemusí se zavádět z disku jako na PC s von Neumannovou architekturou)

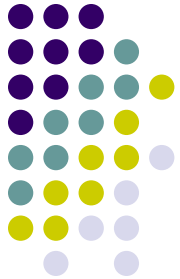
Harvardská architektura



- U harvardské architektury není potřeba mít paměť stejných parametrů a vlastností pro data a pro program
- Paměti mohou být naprosto odlišné, mohou mít různou délku slova, časování, technologii a způsob adresování
- Největší odlišnost pak spočívá v tom, že paměť dat je vždy typu RWM (RAM), zatímco paměť programu bývá nezávislá na napájení (ROM, EEPROM, FLASH)

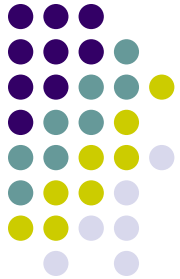
- V moderním počítači typu PC se později setkáme s rozdělením paměti pro data a program v případě cache pamětí uvnitř mikroprocesoru – architektura počítače PC jako celku ovšem stále odpovídá popisu dle Von Neumanna

Srovnání Von Neumannova x Harvardská architektura



- V paměti Von Neumannova počítače jsou uloženy bajty, které jsou daty ke zpracování spolu s bajty, které představují zakódované instrukce programu (strojový kód)
- Skok programu na adresu, kde leží data, způsobí nepředvídatelné chování a havárii – datové bajty bude řadič dekódovat, jakoby to byly operační znaky instrukcí programu
- Dalším problémem je fakt, že **program může přepsat sám sebe** zápisem na adresu, kde je uložen jeho strojový kód
- Von Neumannův počítač **umí naprogramovat „sám sebe“** – na počítači lze psát programy, zkompilovat je spustit (to se na harvardské architektuře nemůže nikdy povést)
- Na počítači Von Neumannovy architektury lze **zavádět programy z disku** (můžete si spustit co chcete), což na harvardské architektuře není možné
- Harvardská architektura má oddělenou paměť programu a tím pádem program nemůže přepsat sám sebe a také nemůže nastat záměna datových bajtů se strojovým kódem, který leží v samostatné paměti dat
- Na harvardské architektuře ale nelze programovat. Program musí být zaveden odněkud „zvenčí“
- Harvardská architektura je ideální pro řídící jednotky, automatizaci, jednoúčelové aplikace...

Jazykový koutek



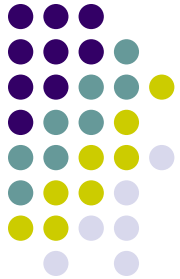
- Psaní pojmů Von Neumannova architektura a harvardská architektura představuje pro technika problém
- Asi nejčastější a nejhloupější chybou je psaní Harvardské architektury s písmenem w jako Har**w**ardské
- Architektura je pojmenována podle univerzity, kde vznikla a jméno této univerzity je Harvard. Žádná univerzita Harward neexistuje, ačkoliv to vypadá „americtější“...
- ..a tuto chybu najdeme i v literatuře renomovaných autorů

- U Von Neumanna je problém s „dvěma n“ a s velkými/malými písmeny
- Je třeba říci, že leckterý češtinář se není zcela jistý psaním velkých či malých písmen ve spojení V/von N/neumannova architektura
- Zatímco v anglické literatuře je běžné psát spojení „... **v**on **N**eumann architecture“ v české je častější „...**V**on **N**eumannova architektura“

1.generace počítačů

- Relé nahradily **ELEKTRONKY**
- Přestávají se používat drátové spoje a nastupují tištěné spoje
- Operační rychlost byla kolem tisíce operací za sekundu
- Příkon desítky kW
- Dosud se programuje výhradně ve strojovém kódu, neexistují programovací jazyky a překladače

Elektronky

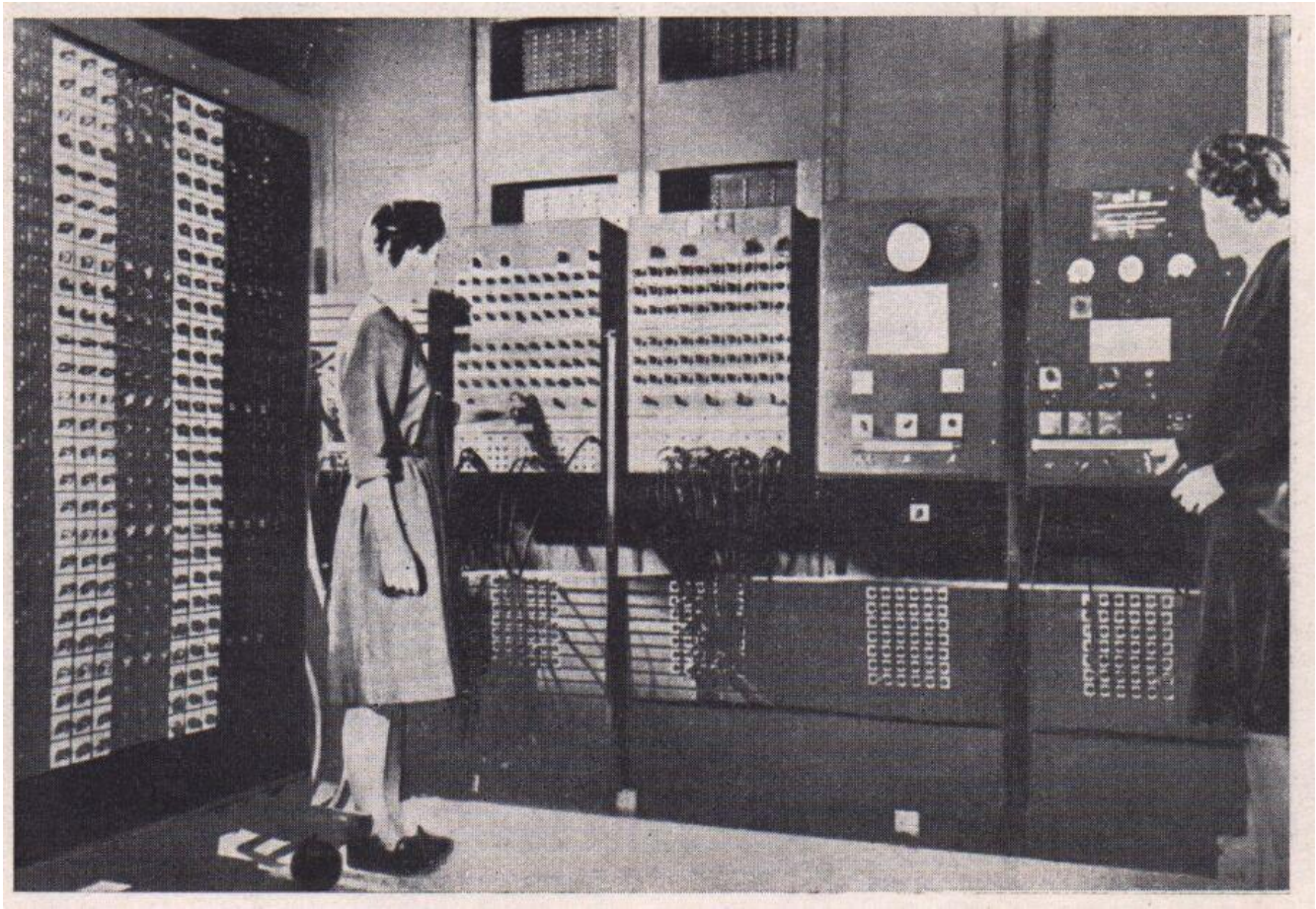


- **Elektronka** je předchůdcem tranzistoru
- Elektronkové počítače byly mnohem rychlejší než předchozí počítače postavené z relé
- Sepnutí a rozepnutí **relé je pomalé** (mechanické kontakty nelze spínat a rozepínat rychleji než cca 100x za sekundu)
- Přepínání bitu 0/1 pomocí elektronky je minimálně tisíckrát rychlejší (nic při tom necvaká, nic se nehýbe, není to mechanické)
- Elektronky jsou ale **velmi poruchové** – elektronková rádia a televize trpěly poruchami téměř každých 10 měsíců
- Počítač obsahující desetitisíce elektronek měl poruchu i několikrát za hodinu
- Elektronka se zahřívá a je energeticky náročná

- Počítače první generace s elektronkami byly **podstatně rychlejší** než počítače nulté generace, ale byly **nespolehlivé** a měli velký elektrický příkon

ENIAC

- Electronic Numerical Integrator And Computer
- První **elektronkový** počítač
- Uveden do provozu v roce 1945

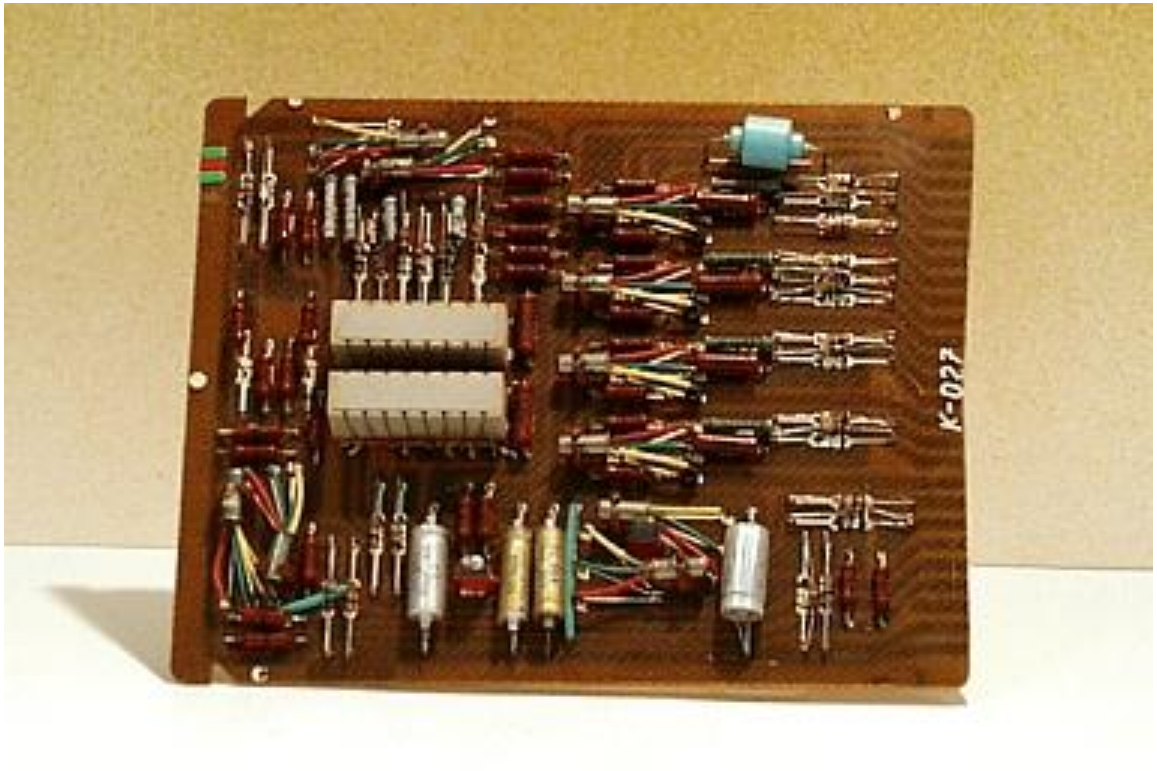


ENIAC

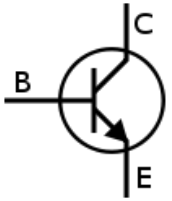
Počítač měl 17 648 elektronek, 1 500 elektromagnetických relé, 70 000 odporů a 10 000 kondenzátorů, které byly s dalšími součástkami umístěny ve čtyřiceti skříních. Měl hmotnost 30 tun, spotřebu elektrické energie 150 kW a zabíral plochu 150 čtverečních metrů. Byl chlazen vzduchem, který hnaly dvě vrtule leteckých motorů

2.generace počítačů (1956-1964)

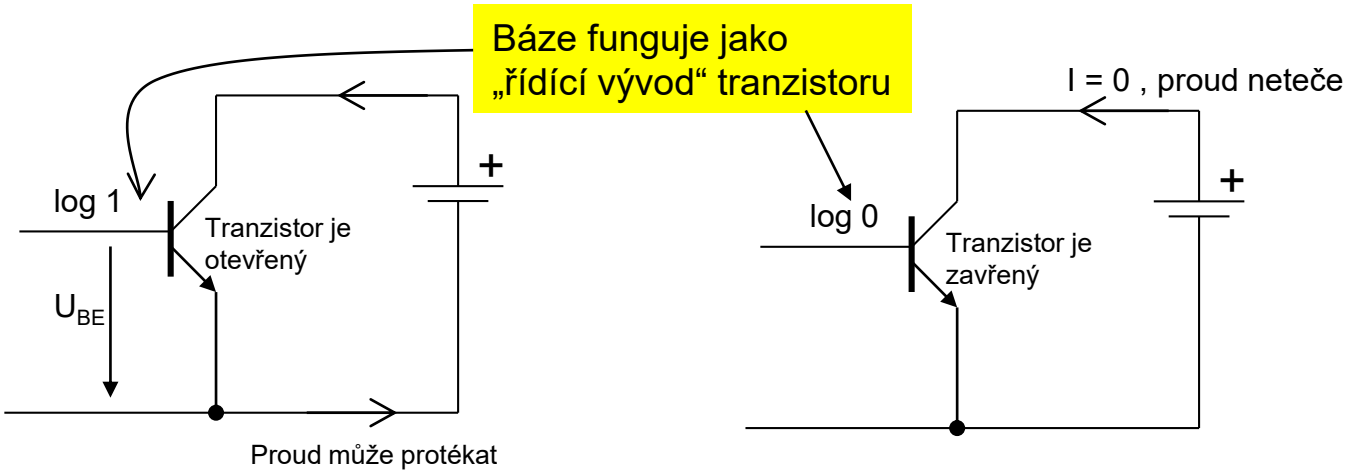
- Místo elektronek se začaly používat **TRANZISTORY**
- Začíná se programovat ve **Fortranu**
- Vynalezeno „**přerušení**“
- Vznikají první **operační systémy**



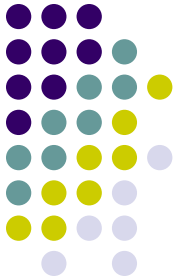
Tranzistor



- **Tranzistor** je polovodičová součástka tvořená dvěma **PN přechody**
- Ve výpočetní technice se tranzistor používá ke spínání – podobně jako dříve relé nebo elektronka
- Tranzistor má tři vývody – báze, kolektor, emitor
- Přivedením logické úrovně 1 na **bázi** tranzistoru se tranzistor **otevře** a může procházet proud mezi kolektorem a emitorem
- Přivedením logické úrovně 0 na **bázi** tranzistoru se tranzistor **zavře** a mezi kolektorem a emitorem nemůže téct proud
- V elektronice se tranzistor používá k **zesilování** – velmi slabý signál přivedený na **bázi** bude zesílen, protože bude silně ovlivňovat (zavírat a otvírat) průchod proudu mezi emitorem a kolektorem

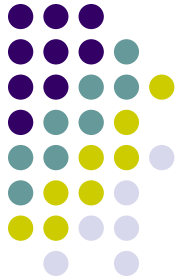


2. Generace počítačů



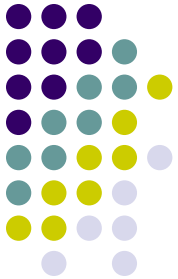
- **Tranzistor** jako základní stavební prvek číslicového počítače je oproti elektronce
  - Mnohem menší
  - Levnější
  - Spolehlivější (je na rozdíl od elektronky prakticky bezporuchový)
  - Rychlejší
  - Má výrazně nižší spotřebu elektrické energie

1956 - přerušení



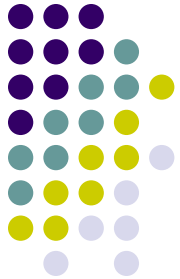
- Rychlost **řadiče a ALU** řádově převyšovala rychlost všech **vstupně/výstupní zařízení** mechanického charakteru (např. čtečka děrných štítků)
- Procesor tato zařízení postupně přestal sám řídit a I/O zařízení byla vybavena vlastním řadičem, ale...
- Většinu času procesor musel trávit **čekáním** na připravenost těchto zařízení (dokončení přenosu, příchod události, čekání na stisk klávesy, periodické testování stavu zařízení)
- Přitom ovšem nemohl dělat něco jiného, užitečného
- I/O zařízení potřebuje mít k dispozici vhodný mechanismus, který mu umožní přihlásit se procesoru a vynutit si jeho pozornost - přinutit jej přerušit provádění právě probíhajícího programu, a místo toho vyvolat jiný program (tzv. obslužný program), který zjistí, co se děje, zareaguje na nastalou událost, a pak se zase vrátí běh přerušenému programu
- Mechanismus přerušení, bez kterého by se dnešní počítače už vůbec neobešly, byl poprvé použit v roce 1956 v počítačích UNIVAC

Přerušení



- **Procesor** provádí výpočet a přitom **čeká** na určitou **událost/signál** (například stisk tlačítka nebo signál od připojeného zařízení)
- Díky **přerušení** nemusí neustále ztrácet čas **testováním**, ale bez starosti o tento signál zpracovává nějaký úkol
- Nastane-li **událost**, na kterou je potřeba nějak zareagovat, dozví se o ní automaticky tak, že zařízení **generuje přerušení** (vyšle přerušovací signál)
- Procesor toto přerušení **obslouží, tzn. přeruší** právě prováděný program a provede nějakou naprogramovanou činnost, kterou má předepsanou provést v okamžiku, kdy nastane událost způsobující přerušení (tomu se říká „**obsluha přerušení**“)
- Po obsloužení přerušení se procesor automaticky vrací ke své původní činnosti
- Procesor je tedy ovládán jak tokem instrukcí v programu, tak **příchodem události**, která způsobí přerušení a provedení předepsané reakce.
- Každé vstupně/výstupní zařízení (klávesnice, síťová karta, pevný disk…) má svůj vlastní přerušovací signál a svou vlastní obsluhu

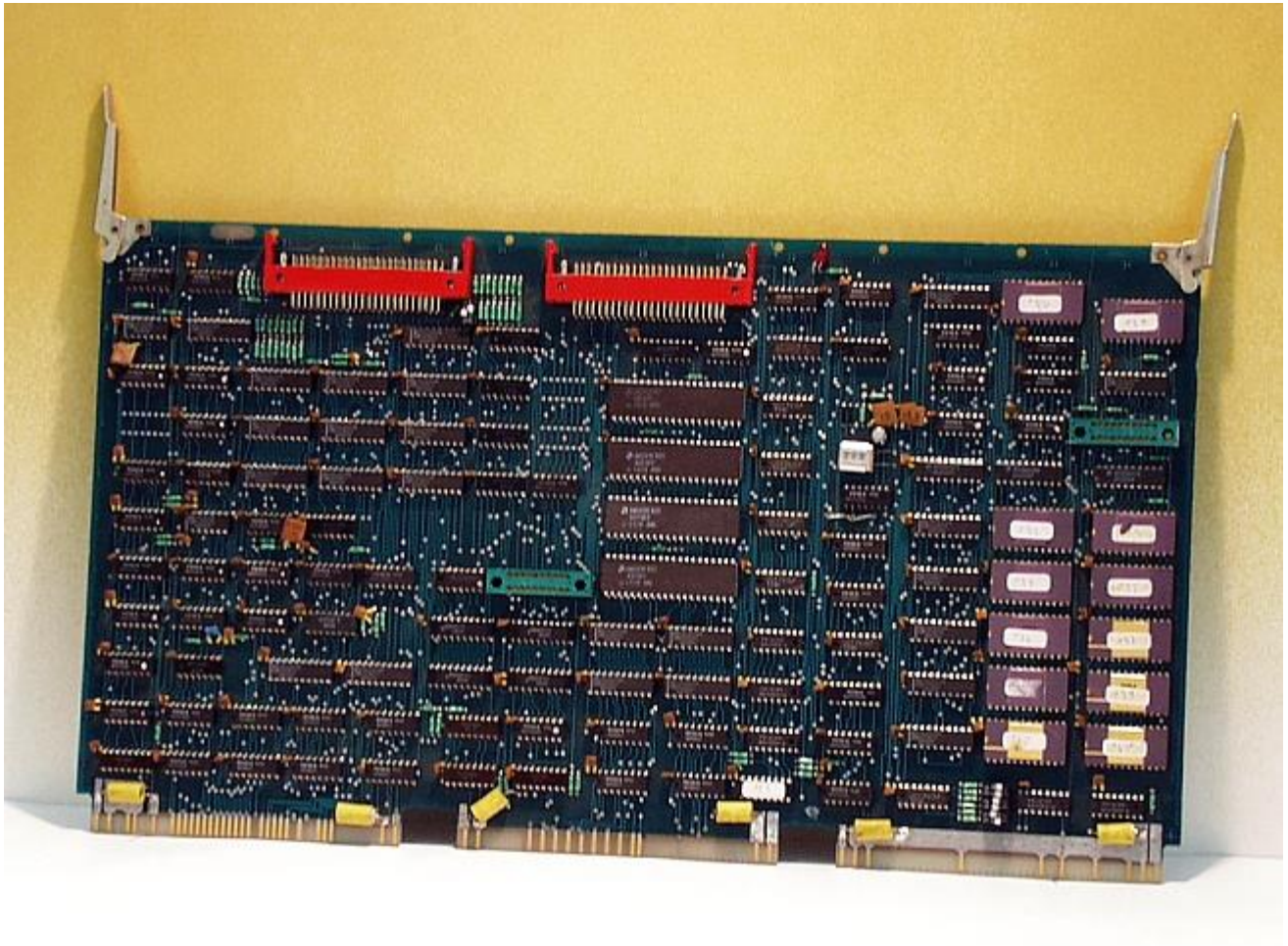
1957 - FORTRAN



- Každý řadič (mikroprocesor) má vždy definován určitý repertoár strojových instrukcí, které je schopen provádět neboli instrukční soubor (instrukční sada)
- každý řadič má obecně jiný instrukční soubor, jiný repertoár registrů, se kterými tyto instrukce pracují, jiné způsoby adresování atd.
- Programátor, který programuje na úrovni **strojového kódu**, si tedy musí být vědom, že pracuje na určitém konkrétním řadiči (procesoru), a musí respektovat jeho specifika
- **Vyšší programovací jazyk** - programovací jazyk, který je nezávislý na konkrétním instrukčním souboru procesoru, a od programátora neočekává znalost konkrétního počítače, pro který bude program psát
- Takový jazyk bude vyžadovat existenci **kompilátoru**, který v něm zapsané programy "přeloží" do strojového kódu
- v roce 1957 skupina vědců z IBM vedená Johnem Backusem vymyslela první vyšší programovací jazyk **FORTRAN**
- **FOR**mula **TRAN**slation
- Dále vznikají jazyky **COBOL** a **ALGOL**

3.generace počítačů

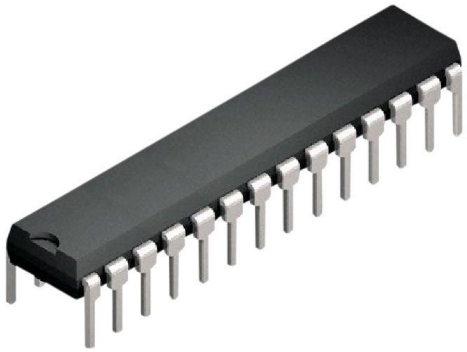
- Integrované obvody
- V roce 1970 dokonce první **MIKROPROCESOR**



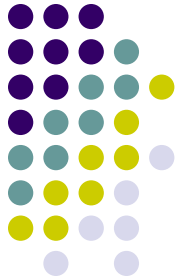
3. Generace počítačů

- Integrovaný obvod je elektronická součástka integrující drobnější součástky (tranzistory, rezistory, kondenzátory, apod.) na jedné polovodičové destičce (obvykle křemíkové) v plastovém pouzdře s vnějšími vývody
- Byl vynalezen ve společnosti Texas Instruments roku 1958

- Výhody
  - Miniaturizace
  - Vysoká spolehlivost
  - Nízká spotřeba el. energie
  - Velmi nízká cena při hromadné sériové výrobě

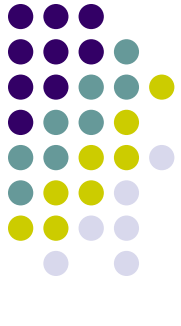


3½ . generace



- V roce 1972 vzniká řada **IBM 370**
- **Integrované obvody** vyráběné technologií MOS dosahují integrace **1000 tranzistorů na čip**
- Vznikají první **magnetické disky**
- Objevují se první **operační systémy**
- První přenosná paměťová média – **8“ diskety** v roce 1971
- Objevují se první „**superpočítače**“ – jejich výpočetní čas se prodává zájemcům o provedení složitých výpočtů
- Počítače se vyrábějí v tisícových sériích
- Jednotlivé typy počítačů jsou naprosto **nekompatibilní** (program napsaný pro jeden počítač nelze spustit na jiném typu počítače)
- Vznikají první **minipočítače** a **mikropočítače** (do té doby byly počítače v podstatě nepřemístitelné obrovské stroje)

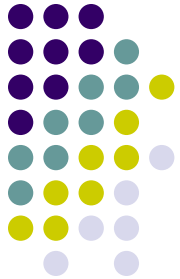
4. generace počítačů



- Počítá se od vzniku **IBM PC/XT** s mikroprocesorem **Intel 8086** – rok 1981
- Přichází éra systému **MS-DOS** a později vznikají také první grafická uživatelská rozhraní (např. Windows, OS2, MacOS, Workbench…)
- Počítače pronikají do **běžných domácností**
- **Cena** počítačů prudce klesá a jejich **výkon** prudce stoupá
- Výrobci se snaží o **kompatibilitu** – program psaný pro jeden počítač by měl být spustitelný i na jiných počítačích

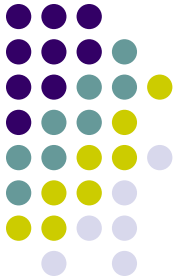
- Je velmi pravděpodobné, že již existují i další generace počítačů, ale zatím nebyly oficiálně stanoveny (dalším zlomovým bodem být mohlo být například propojení Internetem nebo mobilní hardware)
- Delší dobu se experimentuje s počítači na jiném fyzikálním základu.
- Probíhají pokusy s fotonovými počítači a kvantovými počítači – již existují a fungují

Kontrolní otázky



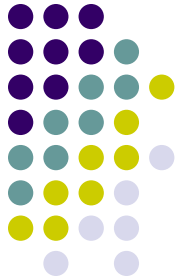
- Popište pravdivostní tabulkou logické obvody AND, OR, NAND, NOR, XOR
- V čem spočívá rozdíl mezi analogovým a digitálním signálem ?
- Kolik různých kombinací lze sestavit ze 14 bitů ?
- Kolik bitů je třeba pro uložení hodnoty 51092 v binárním kódu ?
- Převeďte číslo (1010000000000) z dvojkové soustavy do desítkové a šestnáctkové.
- Co je to strojový kód ?
- Jakým způsobem lze zakódovat číslo -2,049 ? Vymyslete nějaký vlastní 16 bitový kód pro uložení reálných čísel a předveďte, jak byste tuto hodnotu zakódovali.
- V čem spočívá rozdíl mezi ASCII a Unicode ?
- Převeďte do dvojkové soustavy číslo 1025. Jak to lze udělat nejrychleji ?
- Bajt má nulový pátý a šestý bit, ostatní bity jsou nastavené v logickém stavu 1. Zapište stav bajtu binárně, dekadicky a hexadecimálně.
- Určete stav 3. bitu, nejvyššího bitu a nejnižšího bitu v bajtu s hodnotou 148
- Uveďte příklad nějaké jednobitové informace.
- Kolik bitů lze uložit do paměti s kapacitou 2 kB ?
- Kolik bajtů lze uložit do paměti s kapacitou 2 kB ?
- Kolik bajtů je 1 Mib ?
- Kolik bajtů lze uložit na DVD s kapacitou 4,7 GB ?

Kontrolní otázky



- Kolik bajtů lze uložit do paměti s kapacitou 1 GB ?
- Jakou adresou začíná a jakou adresou končí paměť s kapacitou 4 kB ?
- OS Windows ukazuje, že velikost souboru je přesně 2 MB. Kolik bajtů je v souboru uloženo?
- Kolik bajtů je jeden kibibit ?
- Kolik bajtů obsahuje 1 MB dle terminologie výrobců pamětí a kolik podle výrobců pevných disků ?
- Všech prvních dvacet adres paměti obsahuje bajt se stejnou hodnotou 101, pouze adresa 17 obsahuje bajt s hodnotou nula. Zapište standardní výpis obsahu této paměti.
- Jaké základní stavební prvky jsou typické pro počítače nulté generace ?
- Jaké základní stavební prvky jsou typické pro počítače první generace ?
- Jaké základní stavební prvky jsou typické pro počítače druhé generace ?
- Jaké základní stavební prvky jsou typické pro počítače třetí generace ?
- Jak se nazývají vývody tranzistoru ?
- Jaké nevýhody má použití elektronky jako základního stavebního prvku počítače ?
- Jaké nevýhody má použití relé jako základního stavebního prvku počítače ?
- Který blok počítače dle Von Neumannovy architektury umí vynásobit dvě čísla ?
- Který blok počítače dle Von Neumannovy architektury umí porovnat dvě čísla ?

Kontrolní otázky



- Který blok počítače dle Von Neumannovy architektury umí dekódovat strojový kód ?
- Který blok počítače dle Von Neumannovy architektury umí provést negaci bitů ?
- Který blok počítače dle Von Neumannovy architektury umí sdělit paměti, jaká adresa se bude číst ?
- Proč Von Neumannova architektura neobsahuje mikroprocesor ?
- Uveďte alespoň dvě nevýhody Von Neumannovy architektury oproti harvardské architektuře.
- Uveďte výhody Von Neumannovy architektury oproti harvardské architektuře.
- Uveďte alespoň dvě nevýhody harvardské architektury oproti Von Neumannově architektuře.
- Uveďte alespoň dvě výhody harvardské architektury oproti Von Neumannově architektuře.
- Jaká architektura počítače je typická pro osobní počítače PC a proč ?
- Jak se jmenoval první vyšší programovací jazyk ?
- K čemu je dobré přerušení (interrupt) ? Pro jakou generaci počítačů je typické jeho zavedení ?
- Co znamená LIFO a FIFO ?
- Do zásobníku byly postupně uloženy bajty 12, 58, 42, 61 (v uvedeném pořadí). Jestliže nyní dojde ke čtení bajtu za zásobníku, jaký bajt bude přečten ?
- Do zásobníku byly postupně uloženy bajty 12, 58, 42, 61 (v uvedeném pořadí). Jaký bajt leží na vrcholu zásobníku a jaký bajt leží na dně zásobníku ?
- Do fronty byly postupně uloženy bajty 12, 58, 42, 61 (v uvedeném pořadí). Jestliže nyní dojde ke čtení bajtu z fronty, jaký bajt bude přečten ?