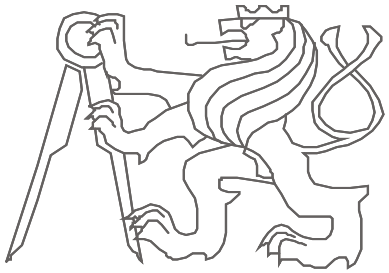
Architektura počítačů

Struktura procesoru a paměti

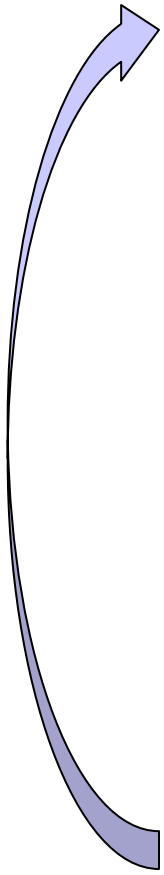
Richard Šusta, Pavel Píša



České vysoké učení technické, Fakulta elektrotechnická

Základní cyklus počítače – sekvenční postup vykonávání instrukcí

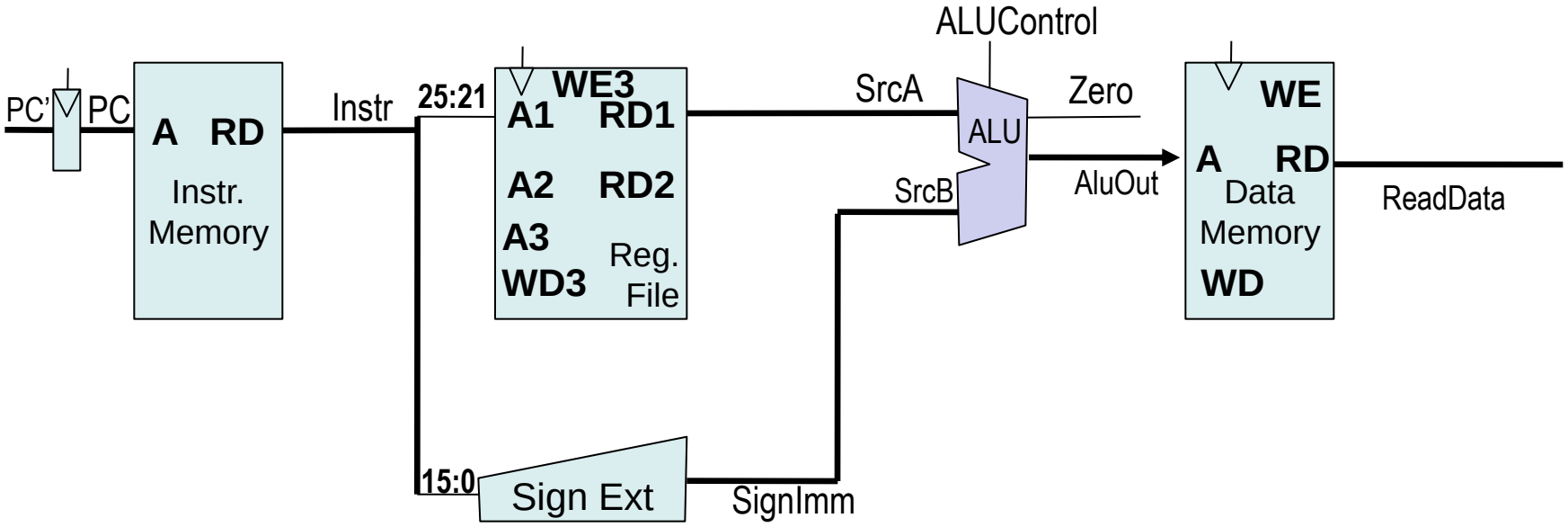
1. Počáteční nastavení, zejména např. PC.
2. Čtení instrukce
   - PC → adresa hlavní paměti,
   - Čtení obsahu,
   - Přečtená data → IR (Instruction Register),
   - PC+n → PC, kde *n* je délka instrukce.
3. Dekódování operačního znaku (OZ),
4. Provedení operace (včetně vyhodnocení efektivních adres, čtení operandů, apod.).
5. Dotaz na možné přerušení. Ano-li, obsluha.
6. Ne-li, opakování od bodu 2.



Jedno-cyklový procesor – návrh – podpora čtení z paměti

- **`lw`**: typ I, rs – bázová adresa, imm – offset, rt – kde uložit

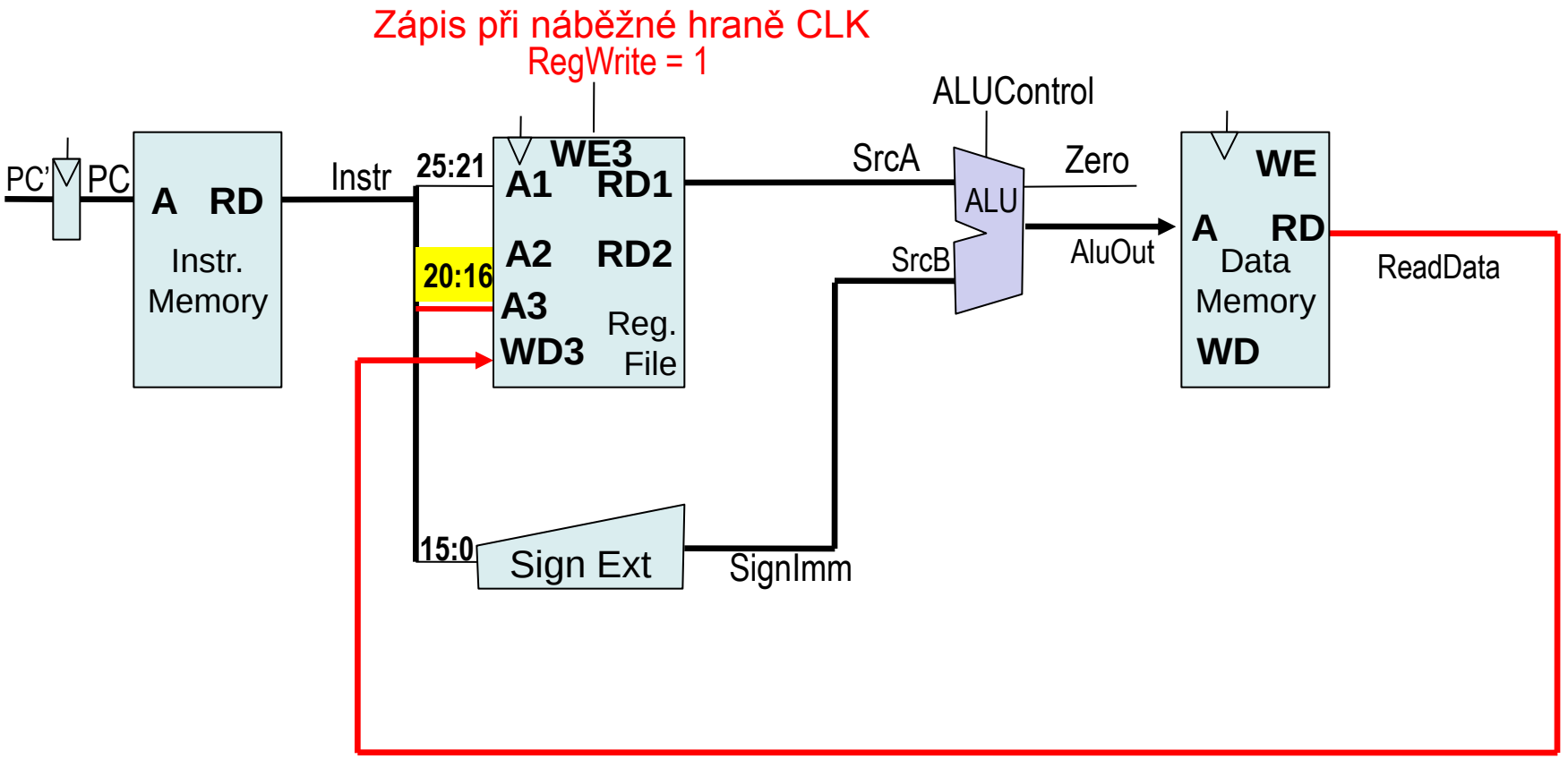
I	**opcode**(6), 31:26	**rs**(5), 25:21	**rt**(5), 20:16	**immediate** (16), 15:0



Jedno-cyklový procesor – návrh – podpora čtení z paměti

- **`lw`**: typ I, rs – bázová adresa, imm – offset, rt – kde uložit

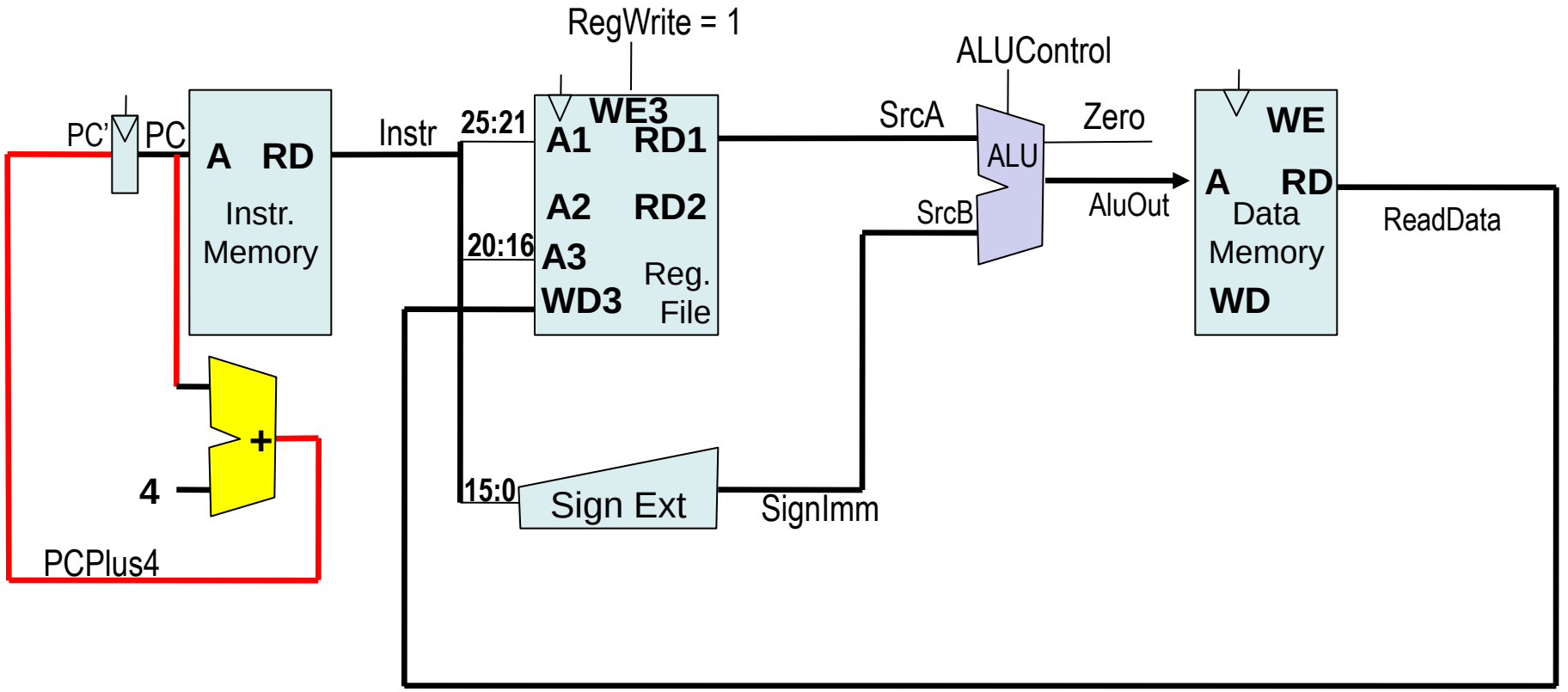
I	**opcode**(6), 31:26	**rs**(5), 25:21	**rt**(5), 20:16	**immediate** (16), 15:0



Jedno-cyklový procesor – návrh – podpora čtení z paměti

- **`lw`**: typ I, rs – bázová adresa, imm – offset, rt – kde uložit

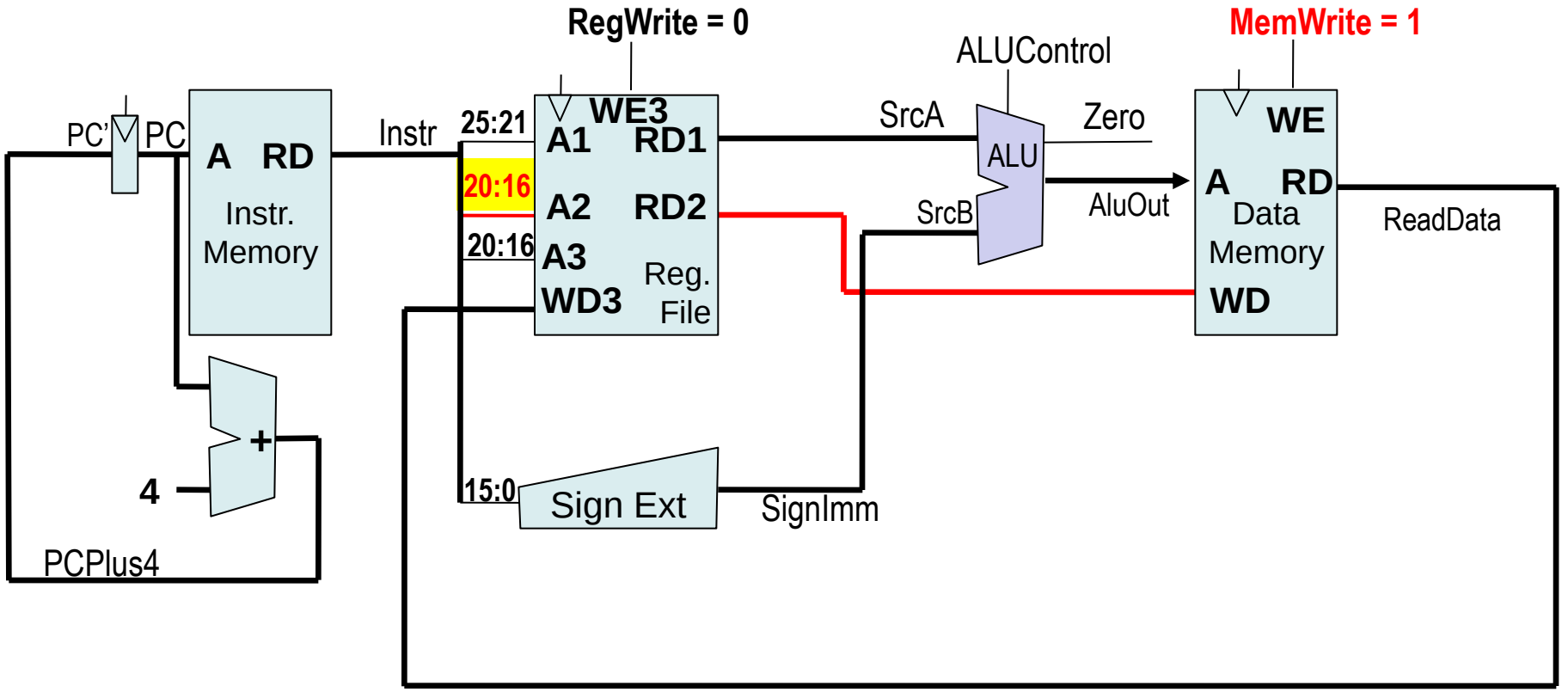
I	**opcode**(6), 31:26	**rs**(5), 25:21	**rt**(5), 20:16	**immediate** (16), 15:0



Jedno-cyklový procesor – návrh – podpora zápis do paměti

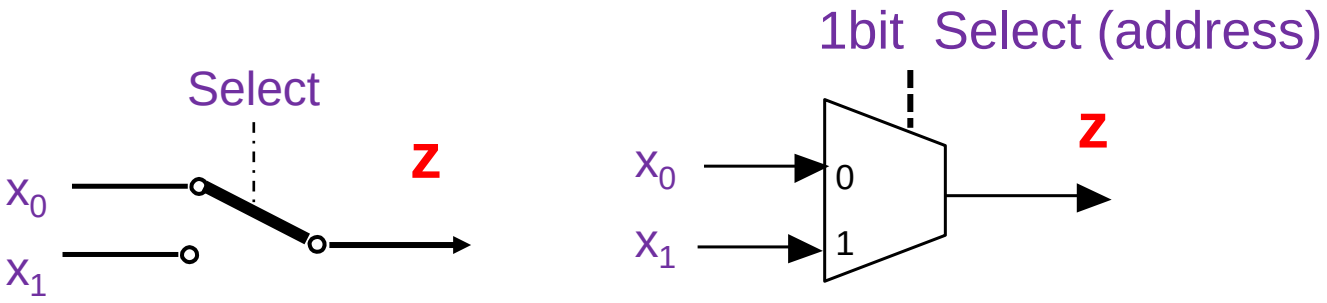
- **`sw`**: typ I, rs – bázová adresa, imm – offset, rt – co zapsat

I	**opcode**(6), 31:26	**rs**(5), 25:21	**rt**(5), 20:16	**immediate** (16), 15:0



Switch analogy

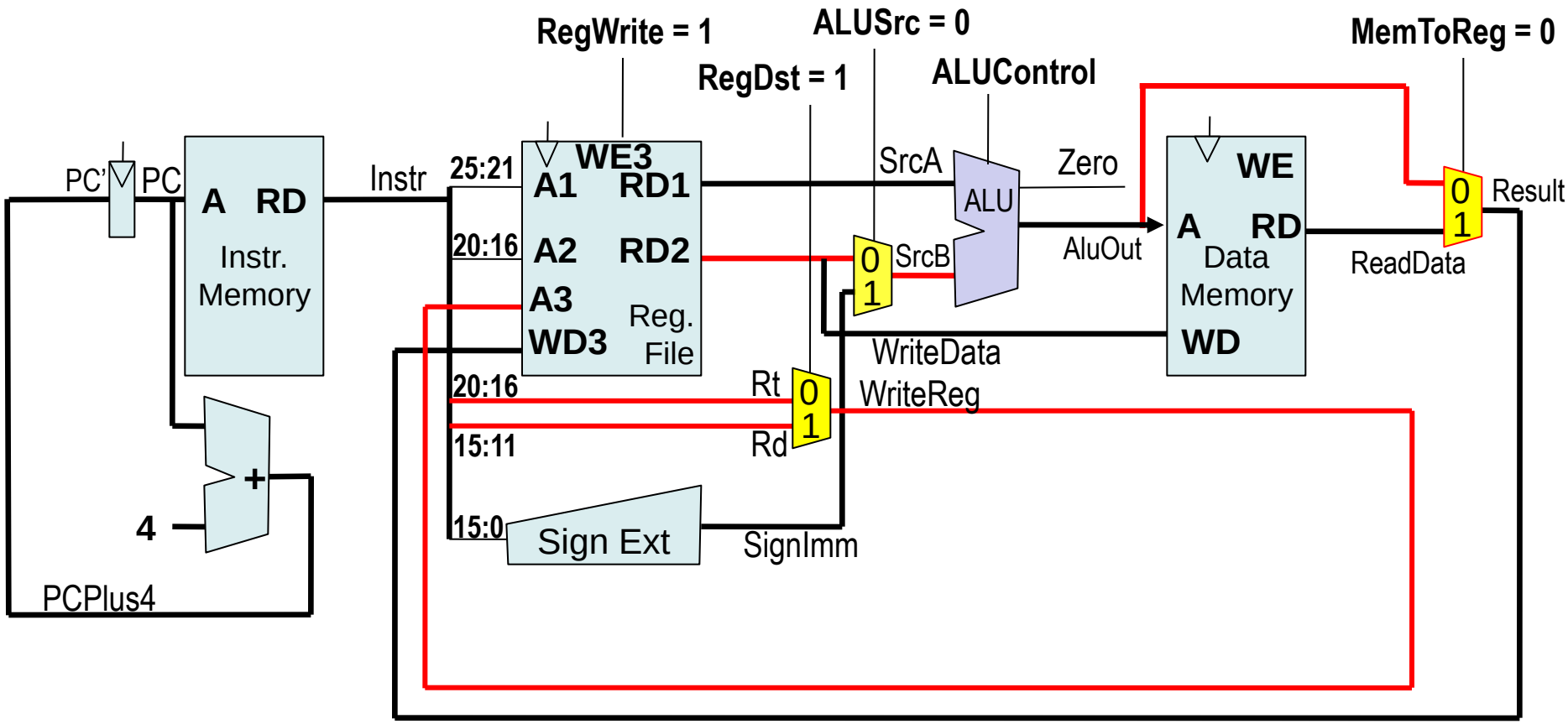
Multiplexer 2 to 1 *cz :2-kanálový (2-vstupový) multiplexor*



Jedno-cyklový procesor – návrh – podpora **add**

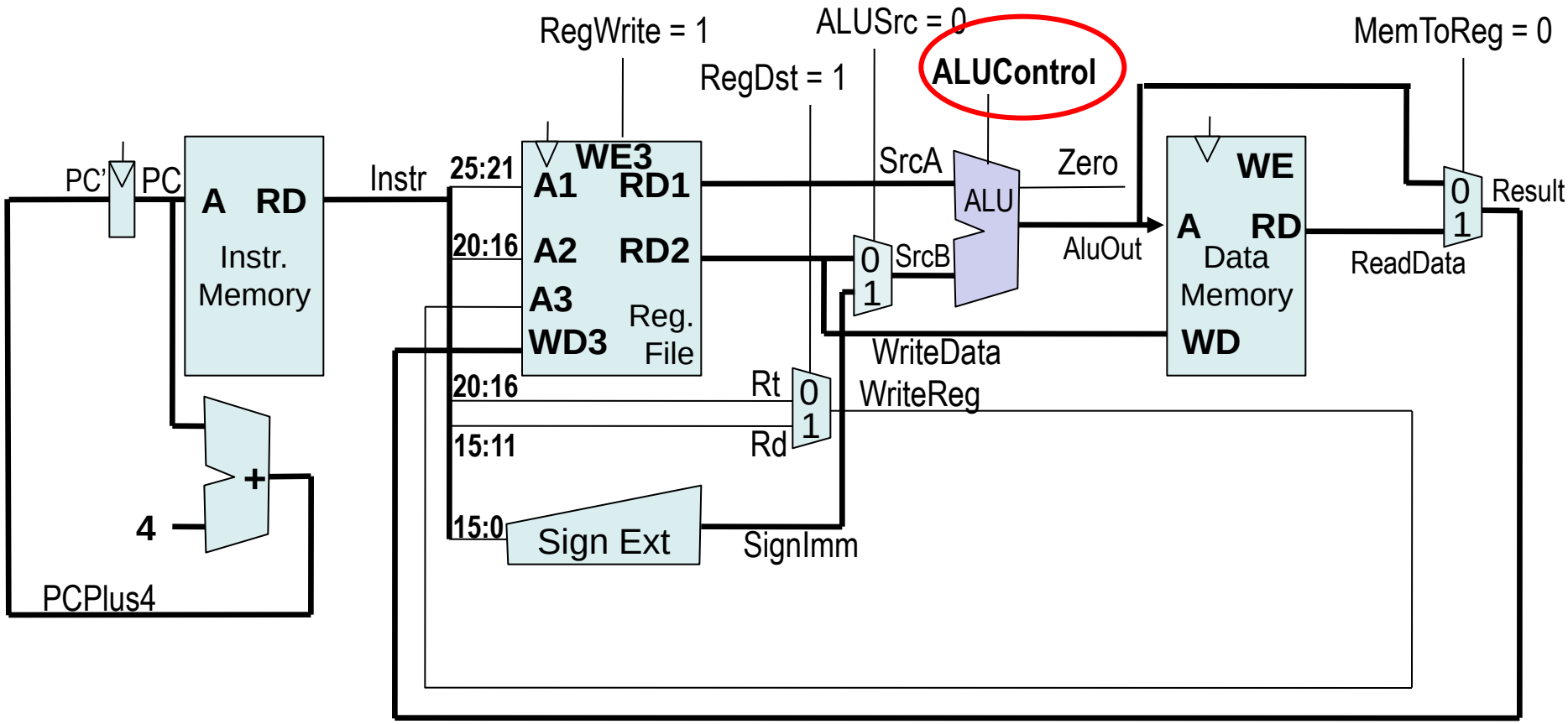
- **add**: typ R; rs, rt – zdroje, rd – cíl, funct – operace součtu

R	**opcode**(6), 31:26	**rs**(5), 25:21	**rt**(5), 20:16	**rd**(5), 15:11	**shamt**(5)	**funct**(6), 5:0



Jedno-cyklový procesor – návrh – podpora *sub, and, or, slt*

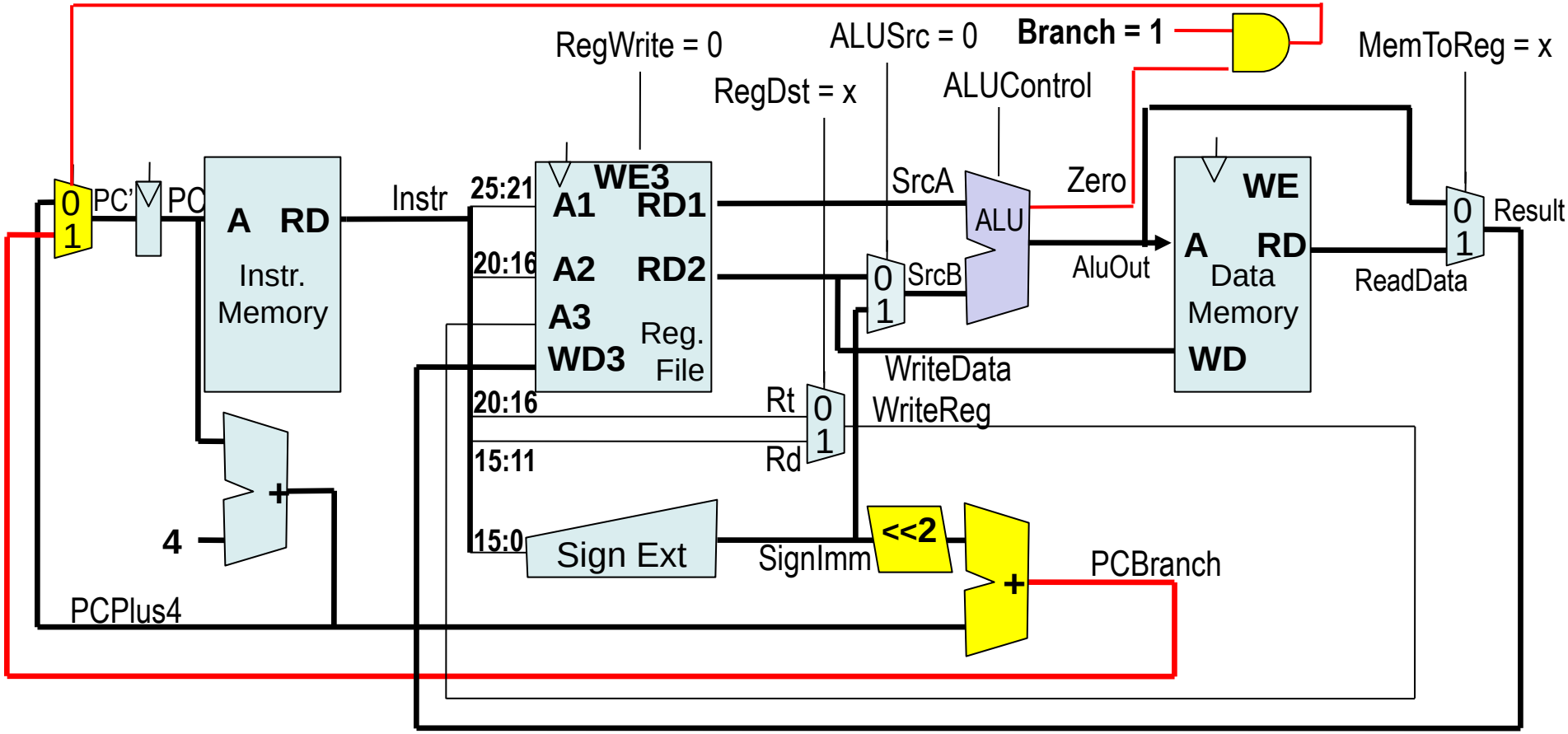
- jediné v čem se liší od `add` je operace ALU -> datapath beze změny; rozdíl v ALUControl



Jedno-cyklový procesor – návrh – podpora **beq**

- **beq** – branch if equal; imm–offset; PC´ = PC+4 + SignImm*4

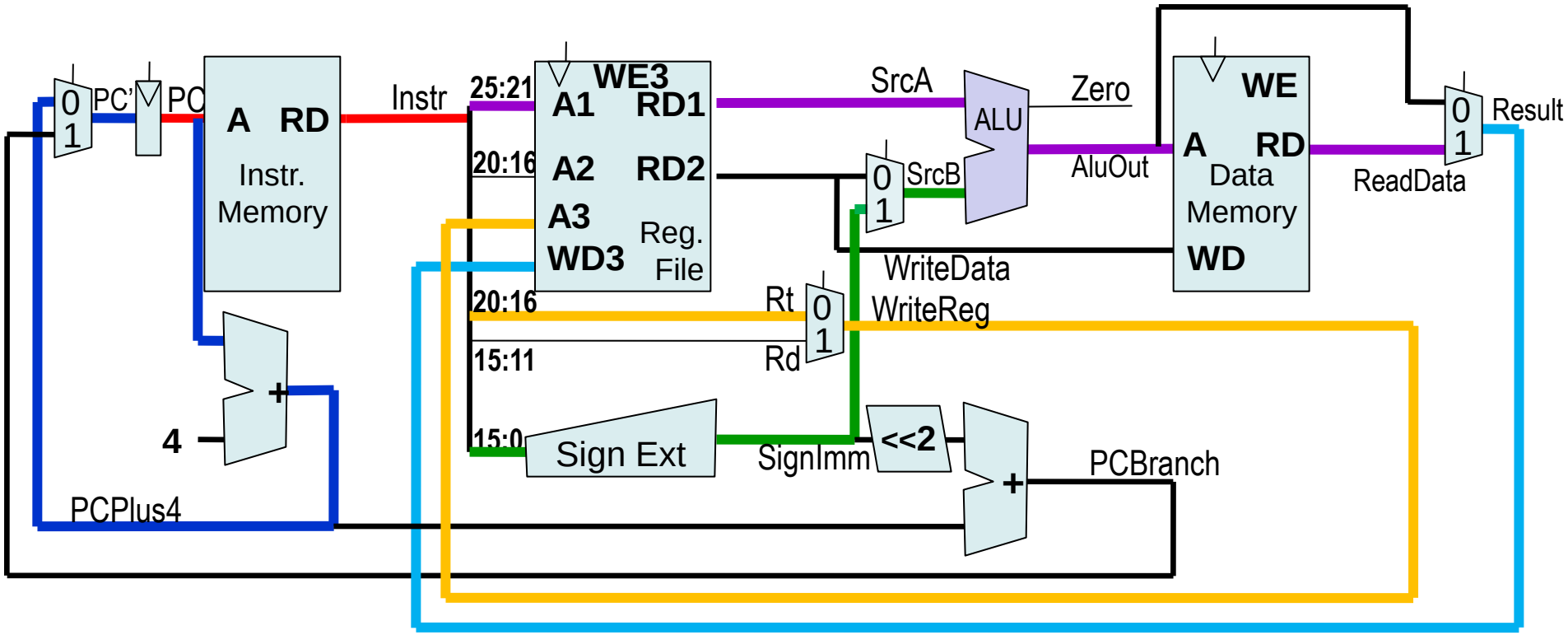
I	**opcode**(6), 31:26	**rs**(5), 25:21	**rt**(5), 20:16	**immediate** (16), 15:0



Jedno-cyklový procesor – výkon: IPS = IC / T = $IPC_{str}.f_{CLK}$

- Jaká může být maximální frekvence procesoru?
- Zpoždění na kritické cestě – instrukce `lw`:

$$Tc = t_{PC} + t_{Mem} + \mathbf{t_{RFread}} + \mathbf{t_{ALU}} + \mathbf{t_{Mem}} + \mathbf{t_{Mux}} + \mathbf{t_{RFsetup}}$$



Jedno-cyklový procesor – výkon: IPS = IC / T = $IPC_{str}.f_{CLK}$

- $Tc = Tc_{instr} + Tc_{proc}$
  $= (t_{PC} + t_{Mem}) + (t_{RFread} + t_{ALU} + t_{Mem} + t_{Mux} + t_{RFsetup})$
- Předpokládejme:

t_{PC}	= 30 ns	t_{Mem}	= 300 ns
t_{RFread}	= 50 ns	t_{ALU}	= 200 ns
t_{Mux}	= 20 ns	$t_{RFsetup}$	= 20 ns

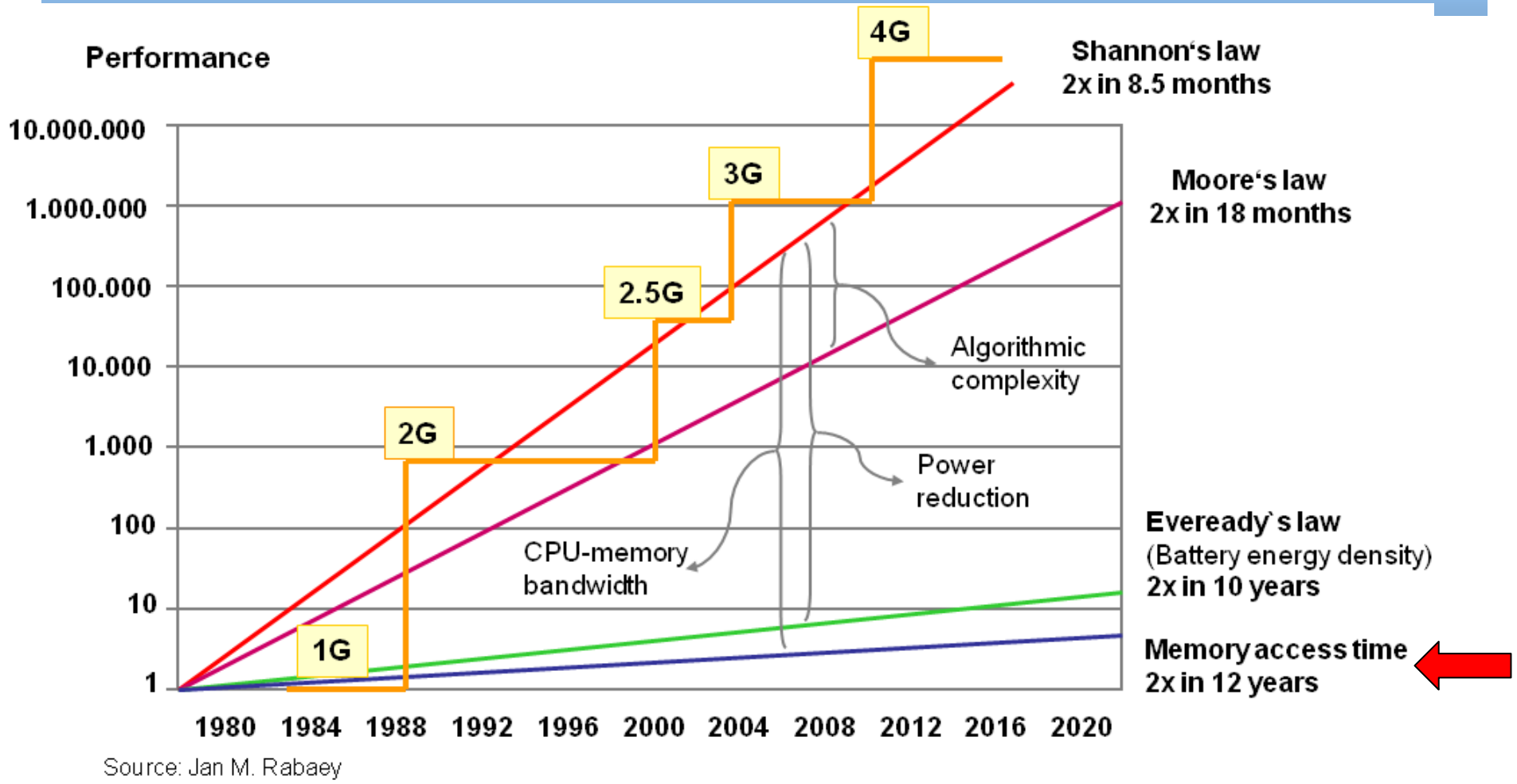
Pak Tc = 920 ns --> $f_{CLK\ max}$ = 1,08 MHz,
IPS = 1 080 000 [instrukce za sekundu]

Ale při Tc_{instr} prováděném paralelně s Tc_{proc},
jelikož je vždy $Tc_i < Tc_p$, pak Tc_p = 50+200+300+20+20
= 590 ns = 1.69 MHz -> **IPS = 1 690 000**

Důležitá poznámka

- Tenhle výsledek si, prosím, zapamatujte.
- Budeme s ním pracovat na pozdější přednášce.

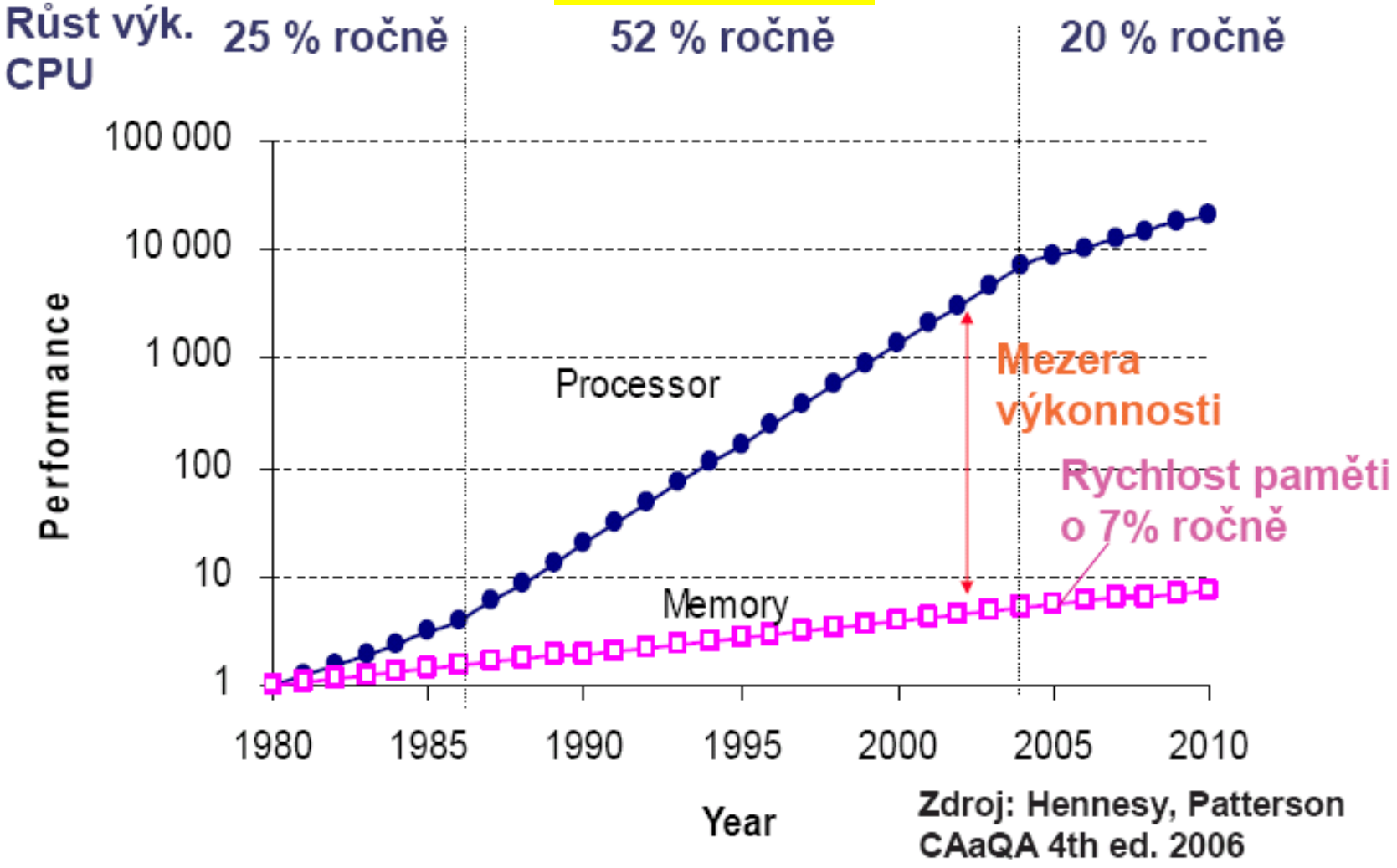
Key Technology Gaps Prediction



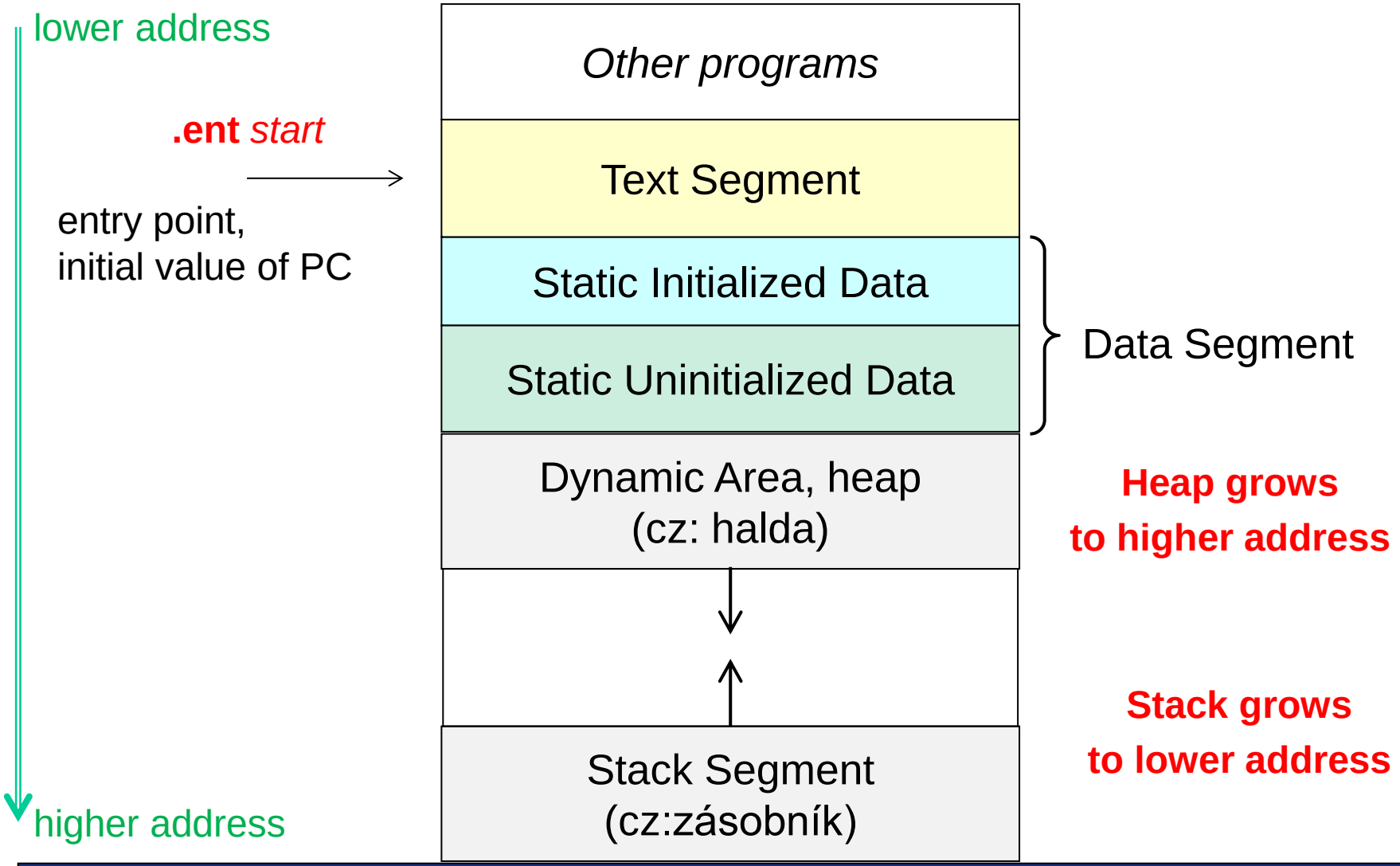
*Note: The increase of algorithm **complexity** over time has been formalized in literature with the so-defined **Shannon's law**.*

Disproporce ve výkonu proc x pam, Moorův zákon

přesnější čísla

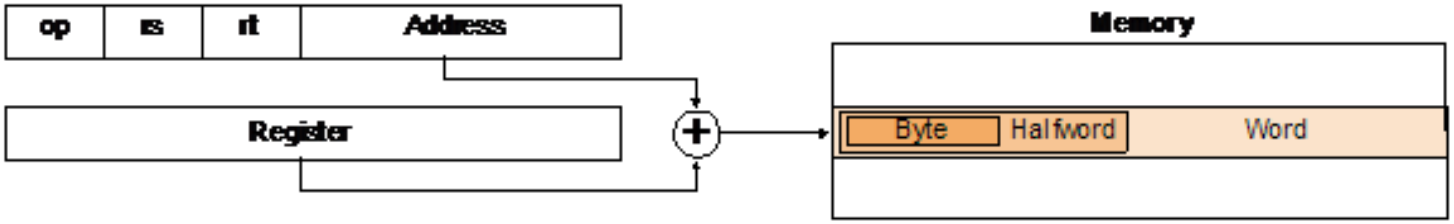


Layout of a Program in Memory



Memory addressing

- Memory address in load and store instructions is specified by a base register and offset



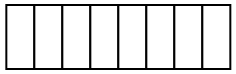
- This is called base addressing

```
addi    a0, zero, 10      // load the word from absolute address
lw     $2, 0x2000($0)     // store the word to absolute address
sw      $2, 0x2004($0)
```

Data Types in Instructions

MIPS registers hold 32-bit (4-byte) words.

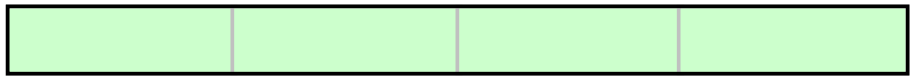
Other common data sizes include byte and halfword.



Byte = 8 bits - example: **lb / lbu**
- load byte extend signed/unsigned

Halfword = 2 bytes example: - **lh / lhu**
- load halfword extend signed/unsigned

Word = 4 bytes - example: - **lw** - load word



u-unsigned extension

Data Directives

.WORD Directive

Stores the list as 32-bit values aligned on a word boundary

.WORD w:n Directive

Stores the 32-bit value *w* into *n* consecutive words aligned on a word boundary.

.HALF Directive

Stores the list as 16-bit values aligned on half-word boundary

.HALF w:n Directive

Stores the 16-bit value *w* into *n* consecutive half-words aligned on a half-word boundary.

.BYTE Directive

Stores the list of values as 8-bit bytes

.BYTE w:n Directive

Stores the 8-bit value *w* into *n* consecutive bytes.

String Directives

.ASCII Directive

Allocates a sequence of bytes for an ASCII string

.ASCIIZ Directive

Same as **.ASCII** directive, but adds a NULL char at end of string

Strings are null-terminated, as in the C programming language

.SPACE n Directive

Allocates space of *n* uninitialized bytes in the data segment

Special characters in strings follow C convention

Newline: \n Tab:\t Quote: \”

Memory Alignment

`.align n` directive

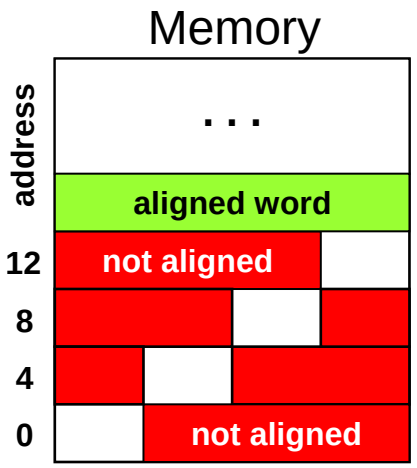
- aligns the next data definition begins on a 2^n byte boundary

.align 2

- the least significant 2 bits of address should be **00**

Memory is addressed as an array of bytes

Words occupy 4 consecutive bytes (MIPS is 32bit processor),



Align

Assembler align data in data segment

```
.DATA
.ALIGN  2
   var1:   .BYTE    3, 5,'A','P','O'
   var2:   .WORD    0x12345678
.ALIGN  3
   var3:   .HALF    1000
```

Example on BIG ENDIAN

var1 ↓ var2 ↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x2000	3	5	41	50	4F				12	34	56	78				
0x2010	10	00														

var3 ↑

Motivace pro další část z pohledu programátora?

Otázka 1. Vykonávají programy to samé?

Otázka 2. Který program je rychlejší (pokud některý)?

A:

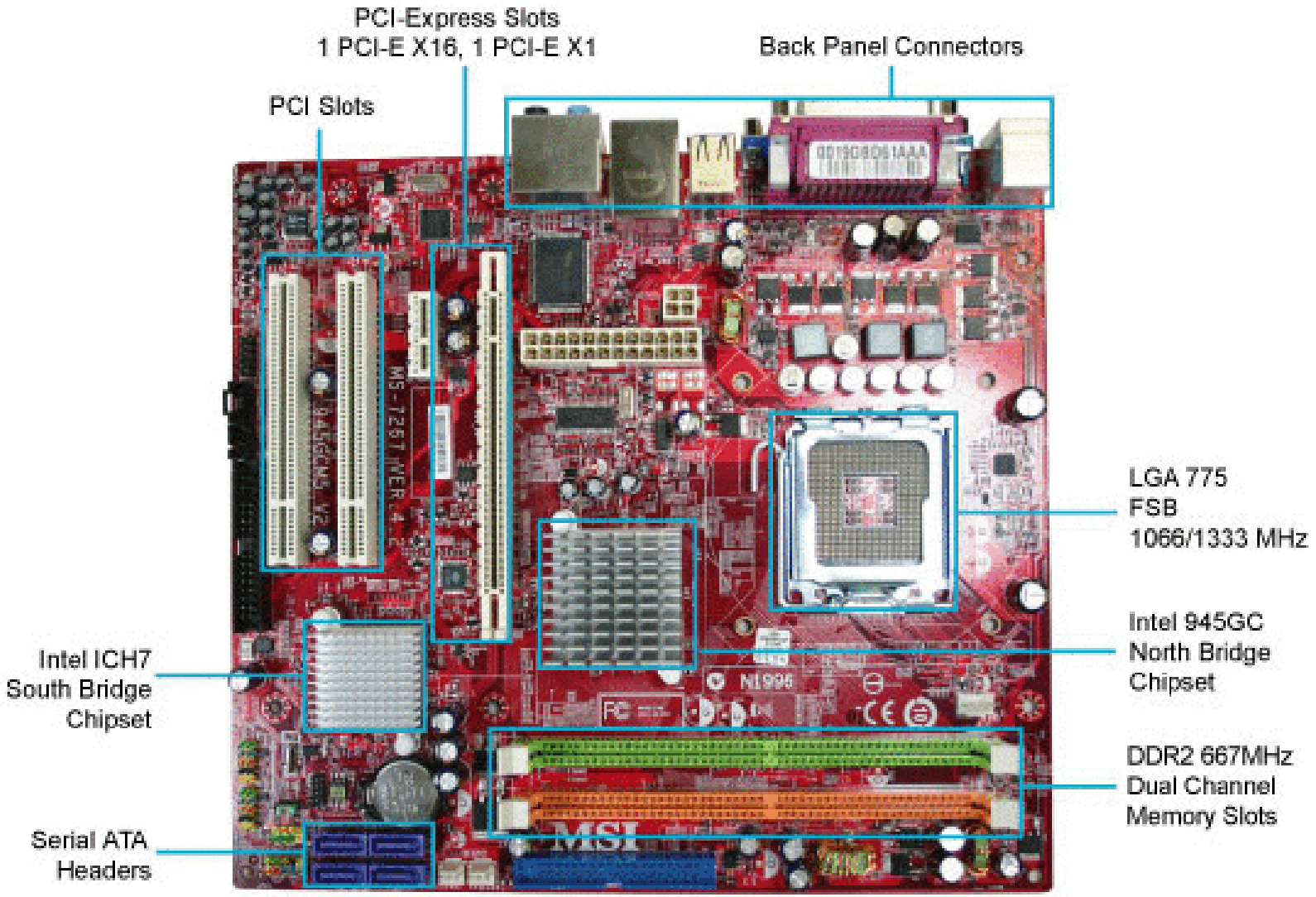
```
int matrix[M][N];
int i,j,sum=0;
…
for(i=0;i<M;i++)
  for(j=0;j<N;j++)
    sum+=matrix[i][j];
```

B:

```
int matrix[M][N];
int i,j,sum=0;
…
for(j=0;j<N;j++)
  for(i=0;i<M;i++)
    sum+=matrix[i][j];
```

Lze doporučit výhodnější způsob procházení matice?
K odpovědi je nutná znalost organizace paměti.

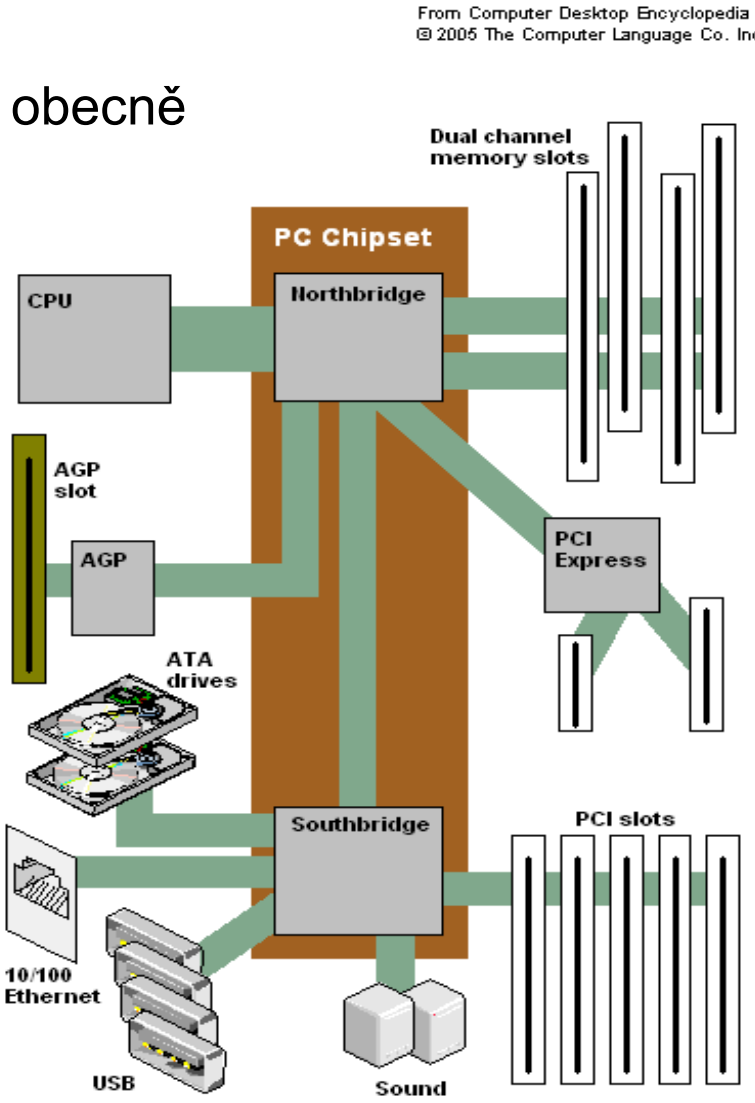
Architektura počítače



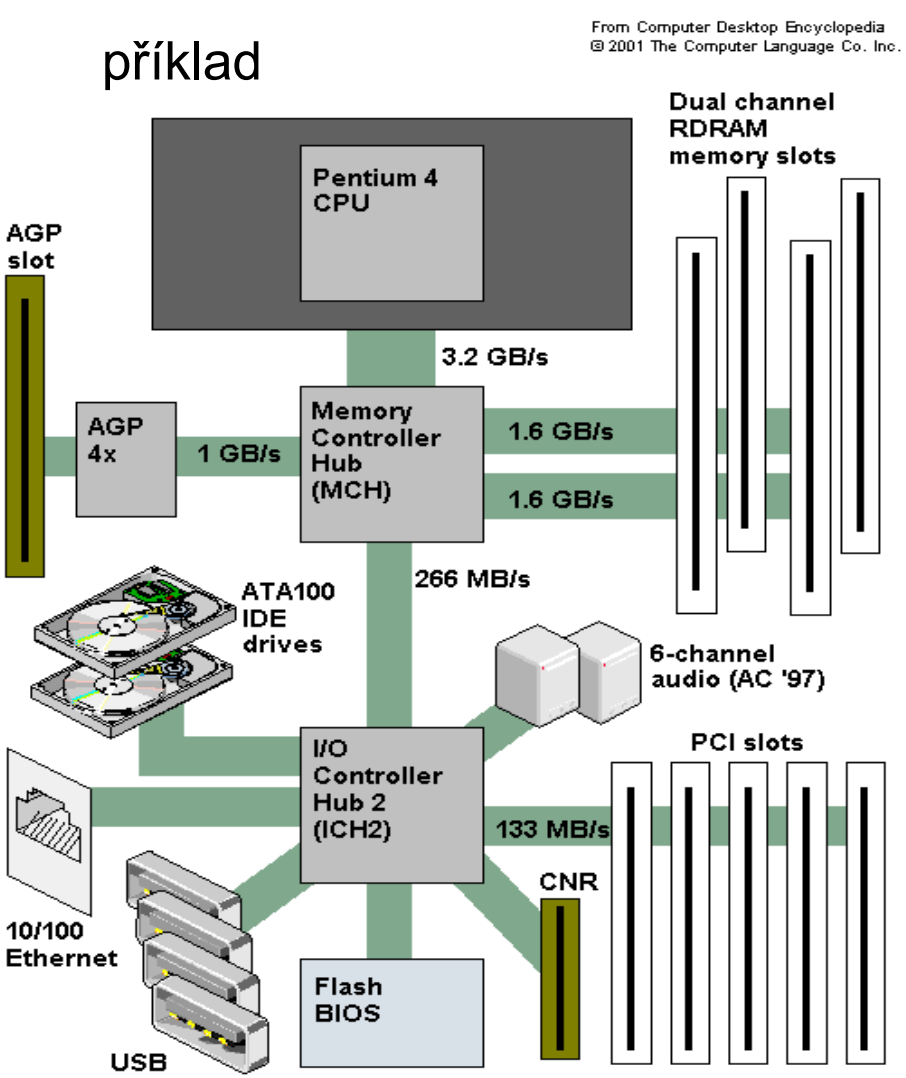
http://www.pcplanetsystems.com/abc/product_details.php?item_id=3263&category_id=208

Architektura počítače

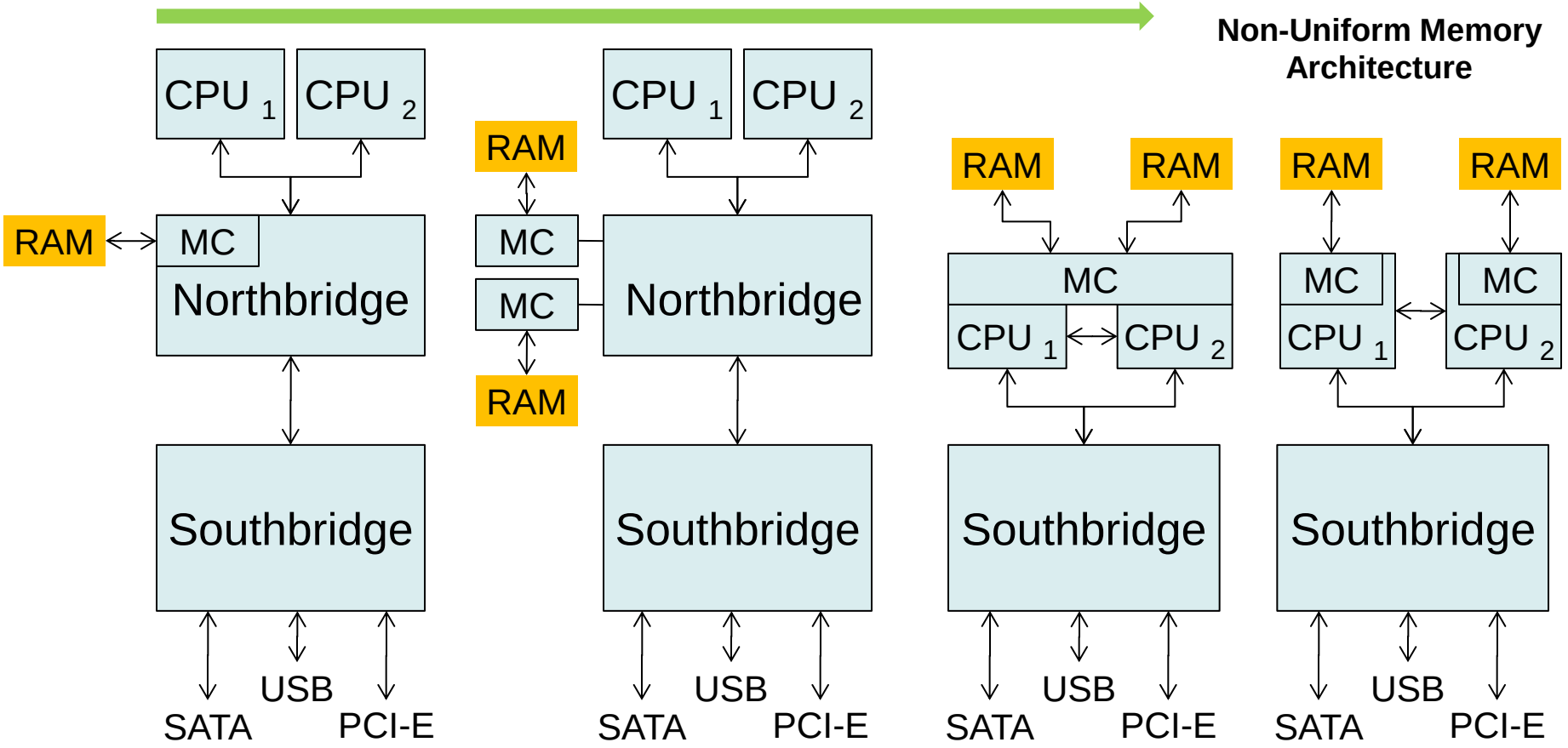
obecně



příklad

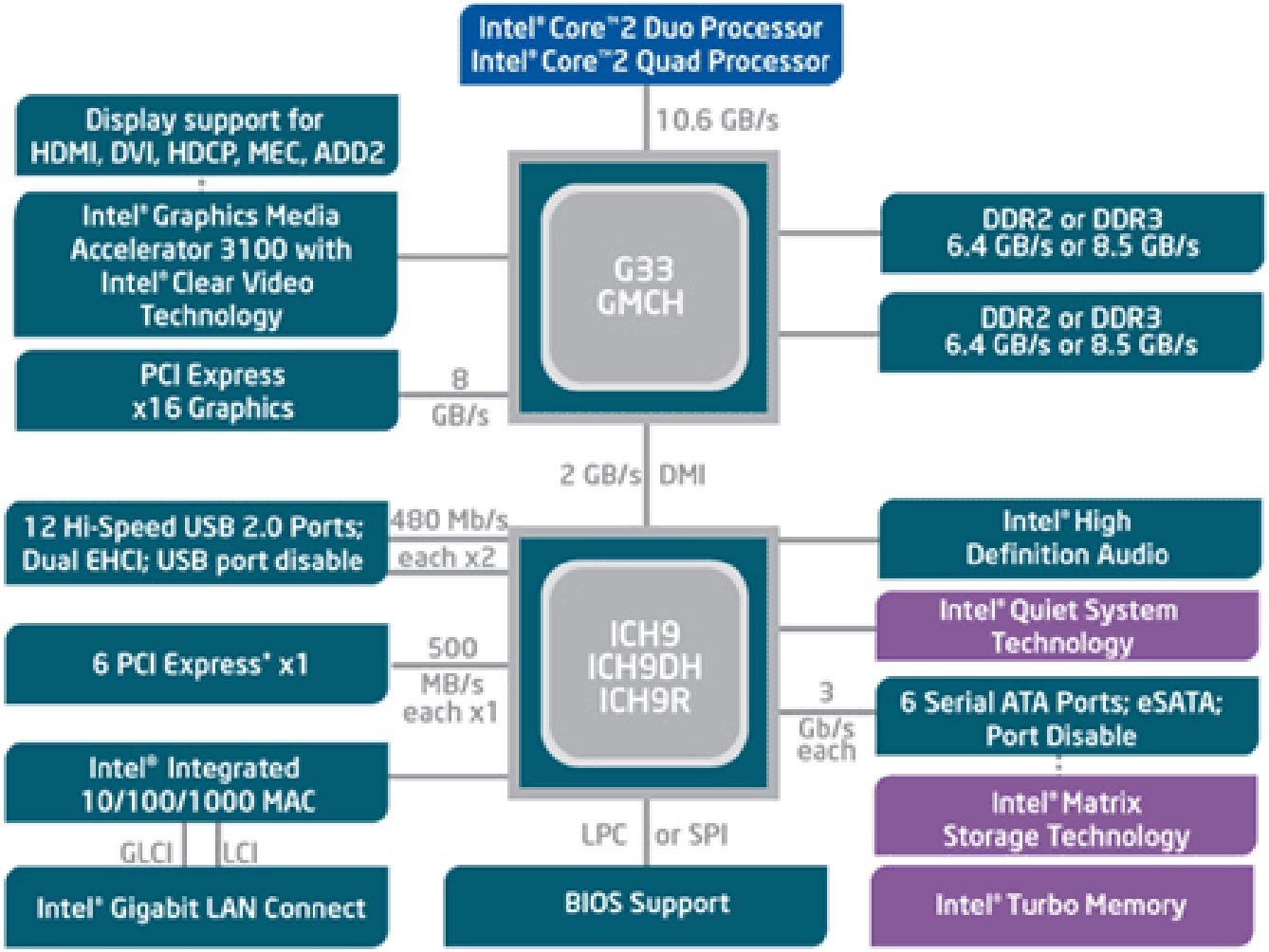


Vývoj architektury počítače



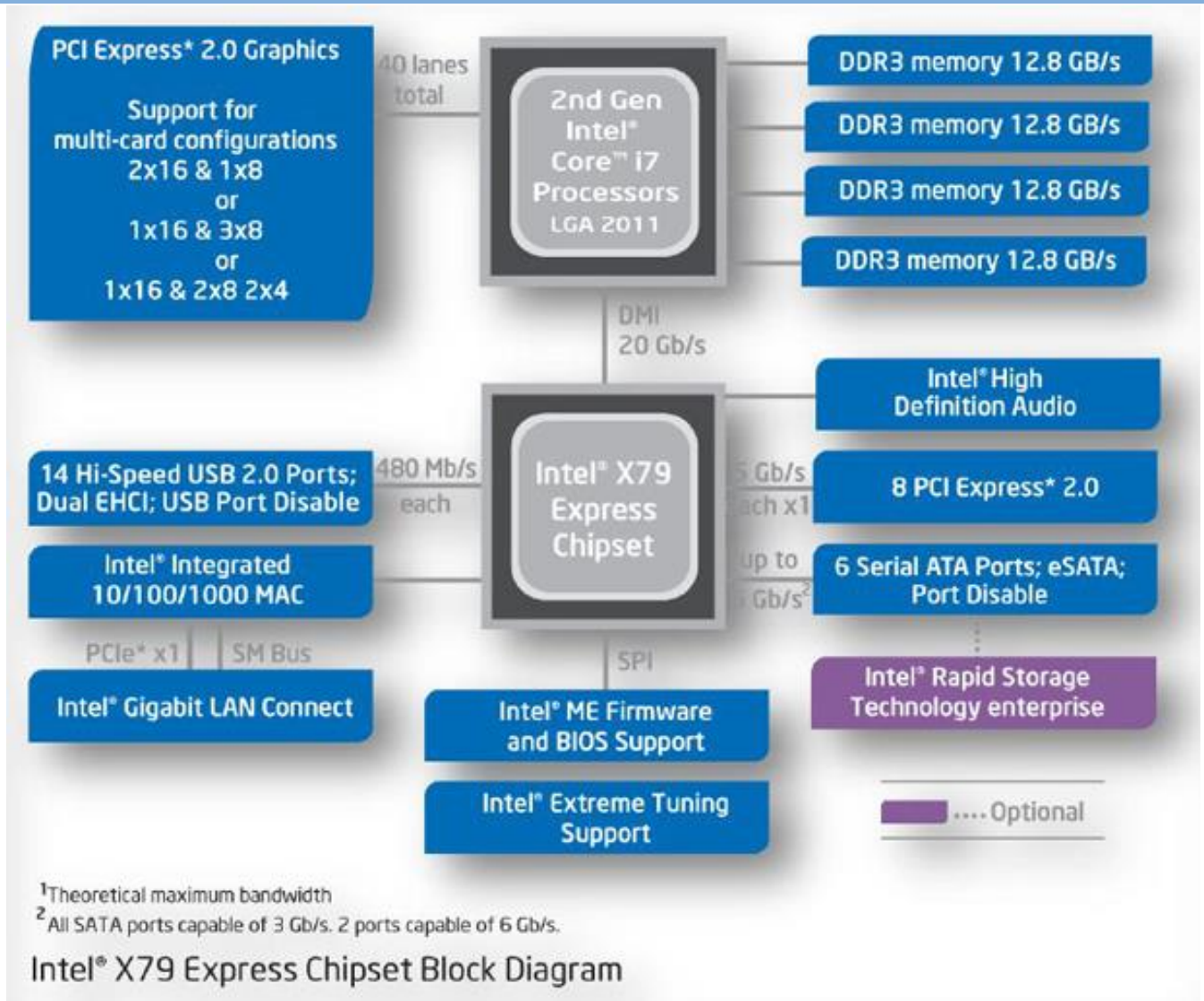
MC - Memory controller *– obsahuje obvody pro zajištění operace čtení a zápisu z/do paměti. Také se stará o udržení obsahu paměti*
– refresh typicky každých 64ms. Některé DRAM umí i self-refresh, který je výhodný, když se z paměti nečte, jinak by mohl blokoval dostup.

Konkrétněji…



Nyní je Northbridge jako Graphics and Memory Controller Hub (GMCH)

Konkrétněji současnost…

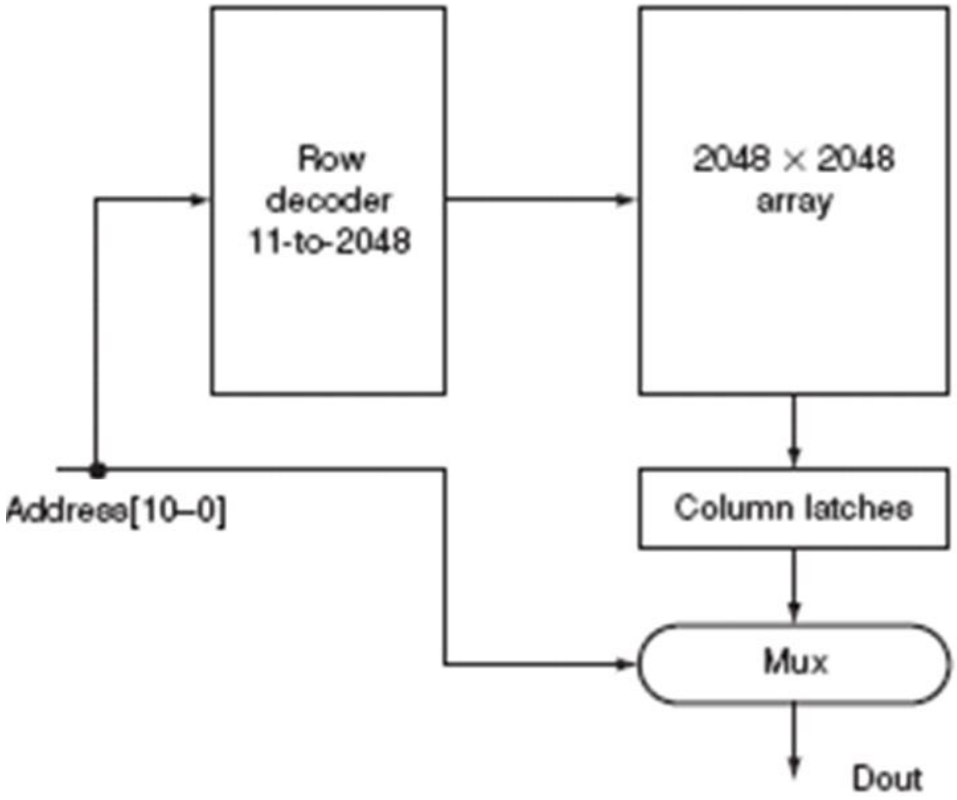


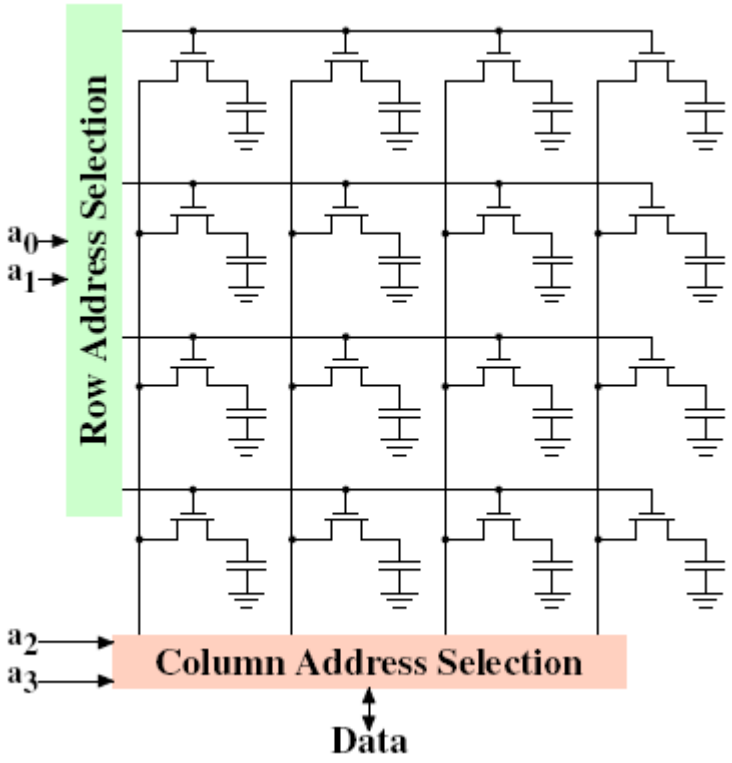
Terminologie kolem pamětí

- **Adresa**, pojem snad není třeba vysvětlovat.
- **Hodnota**, vlastní informace. Paměťová buňka však může obsahovat i další informaci (třeba o platnosti hodnoty, apod.).

- Parametry pamětí:
  - **Vybavovací doba paměti**, kritický parametr. Délka časového intervalu mezi objevením se požadavku a okamžikem, kdy jsou data k dispozici.
  - Doba přístupu, zastaralý parametr; vybavovací doba + obnovení obsahu po destruktivním čtení.
  - **Propustnost**, výkonový parametr. Schopnost zpracovat uvedené množství za jednotku času.
  - Latence = zpoždění, podobně jako vybavovací doba.

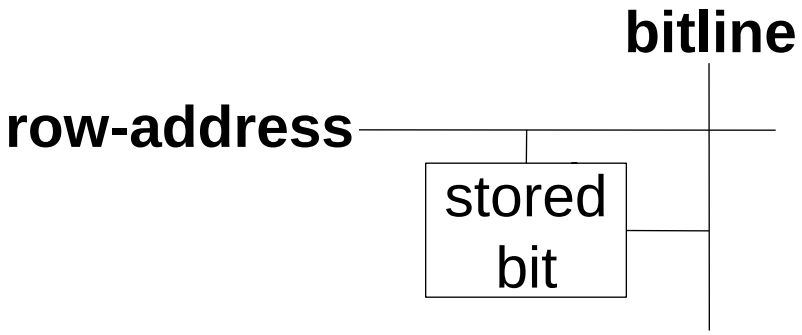
Vnitřní organizace čipu DRAM paměti

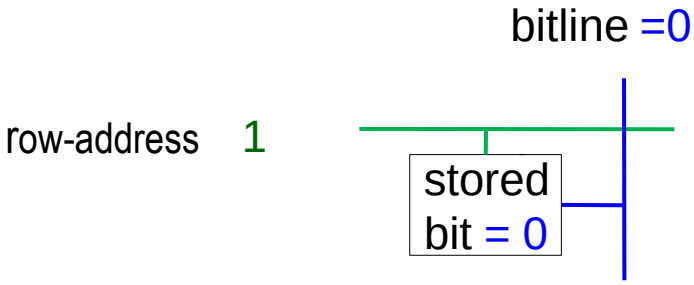


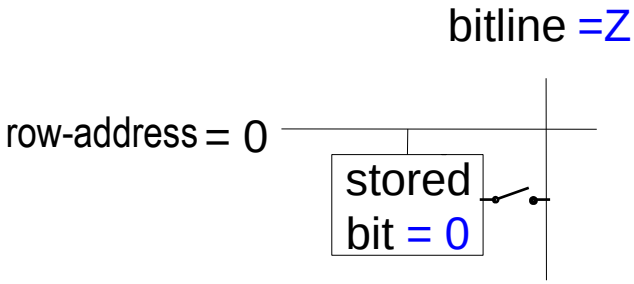


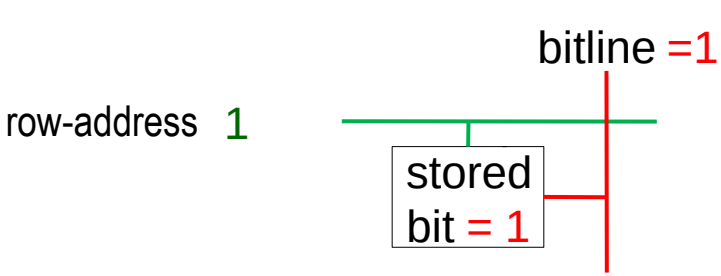
4M × 1 DRAM (Dynamic Random Access Memory)
je uvnitř realizována jako pole 2048 × 2048 1b paměťových buněk

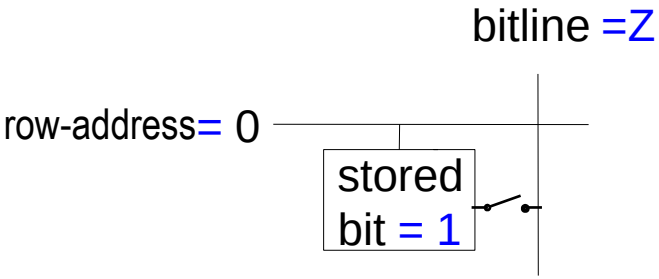
Memory Cell


bitline
row-address
stored
bit


bitline =0
row-address 1
stored
bit = 0

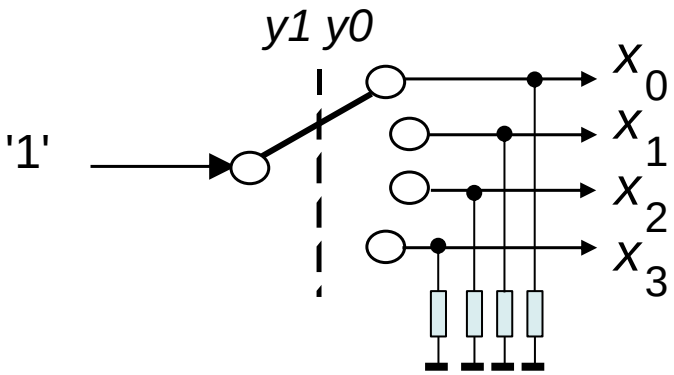

bitline =Z
row-address = 0
stored
bit = 0

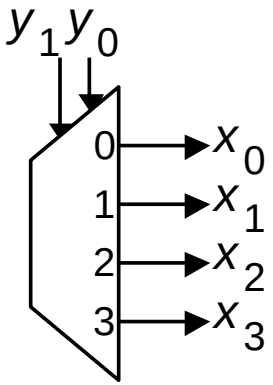

bitline =1
row-address 1
stored
bit = 1


bitline =Z
row-address= 0
stored
bit = 1

Switch Analogy of Decoder

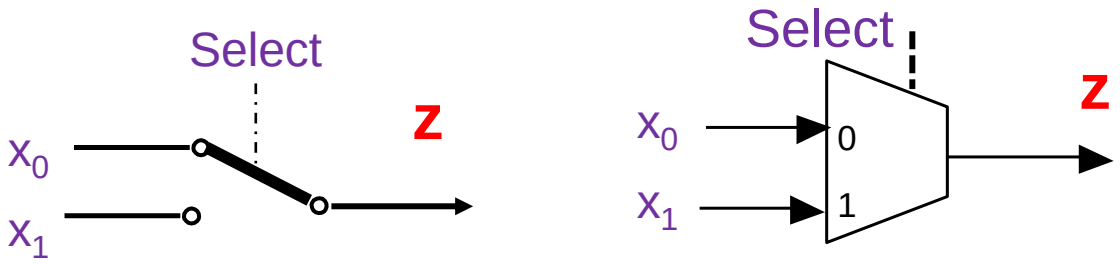
One Hot Decoder *cz: Dekodér 1 ze 4*



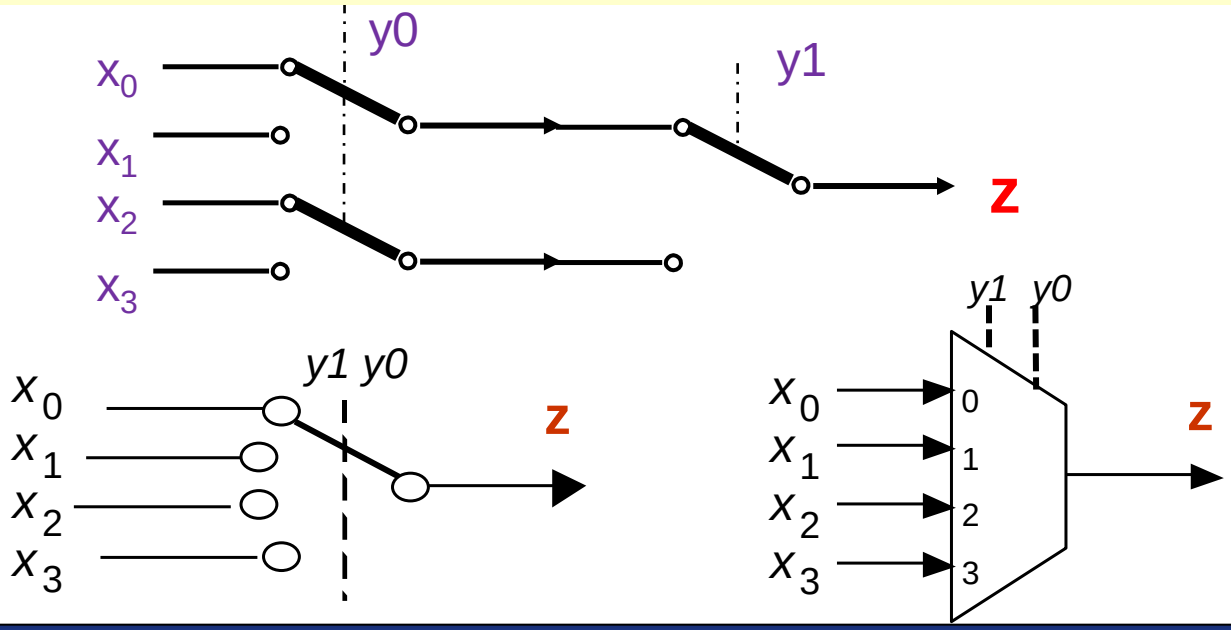


Switch analogy

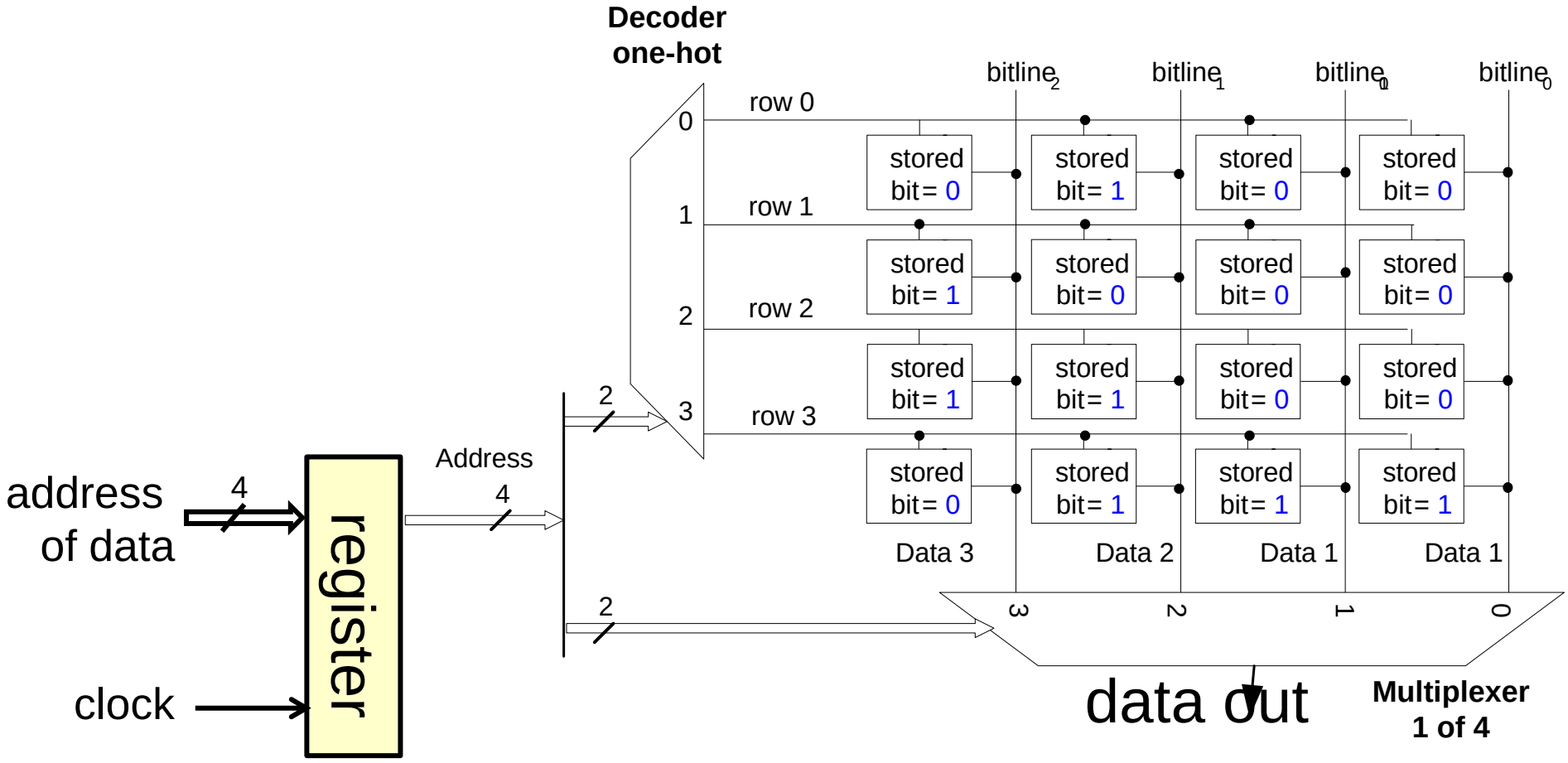
Multiplexer 2 to 1 or **1 of 2** *cz :2 kanálový (2-vstupový) multiplexor*



Multiplexer 4 to 1 or **1 of 4** *cz : 4 kanálový (4-vstupový) multiplexor*

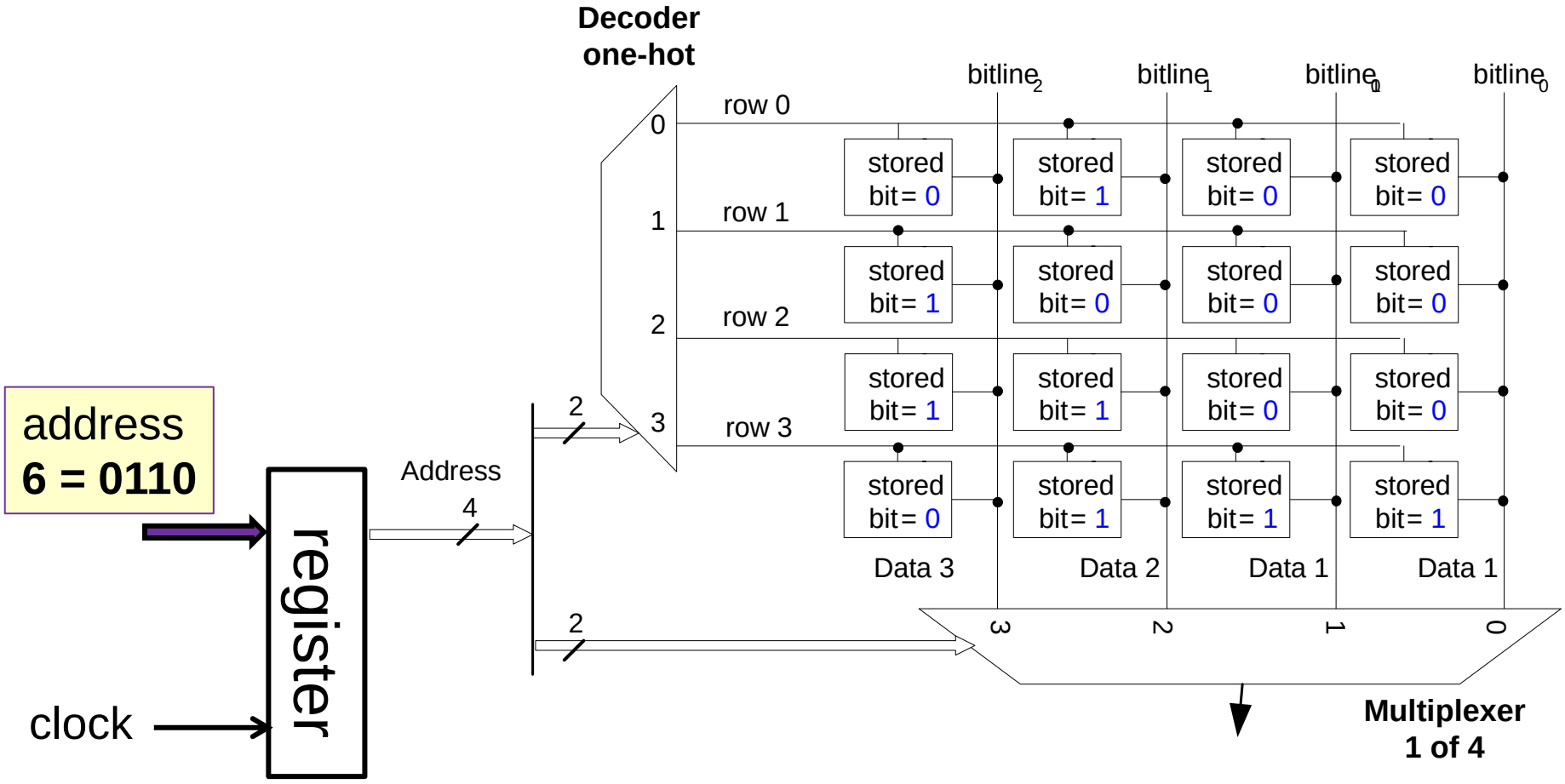


Memory matrix



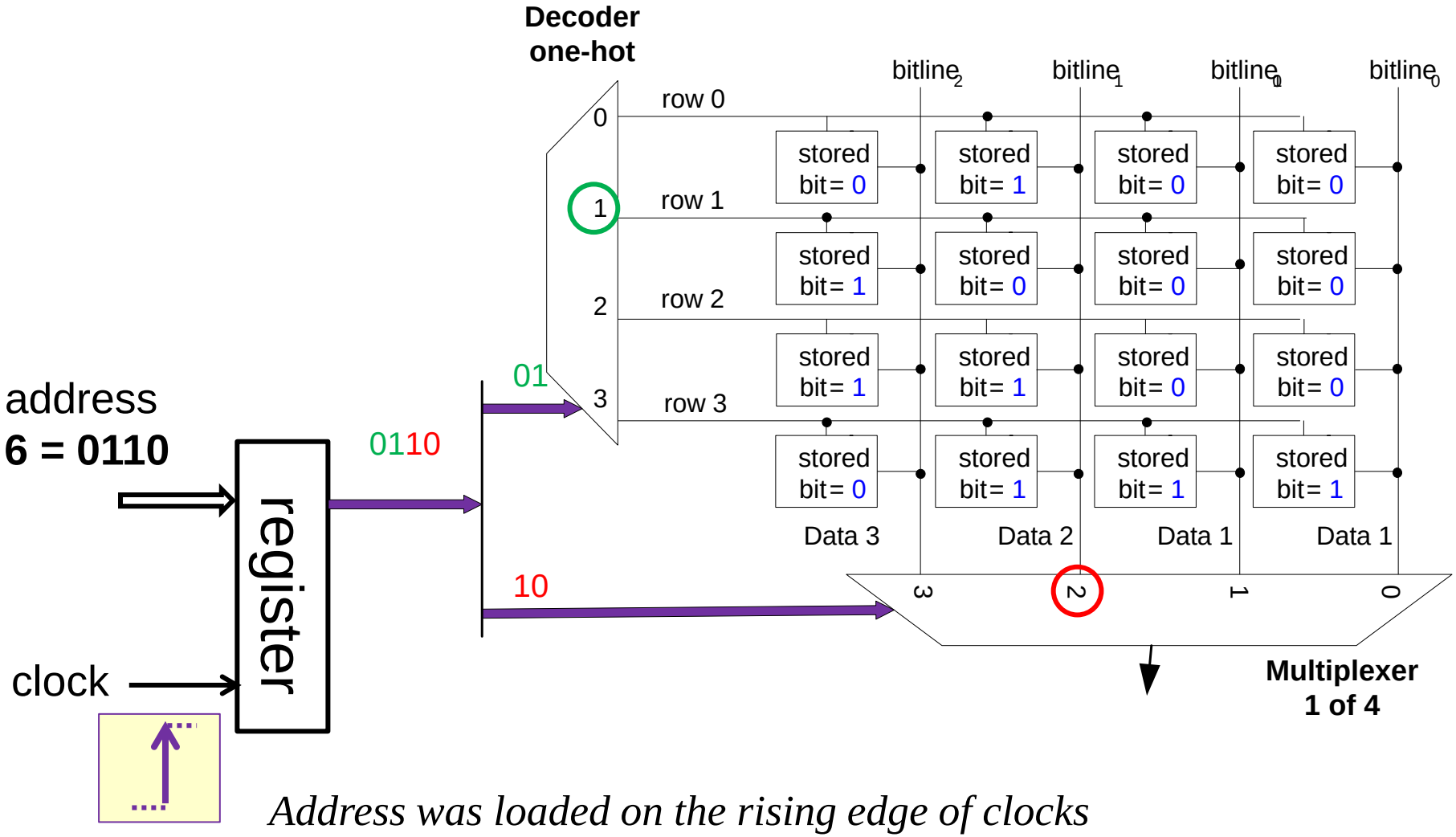
Register of address is a necessary condition for the implementation and to reduce power consumption

Memory matrix - example 1/4



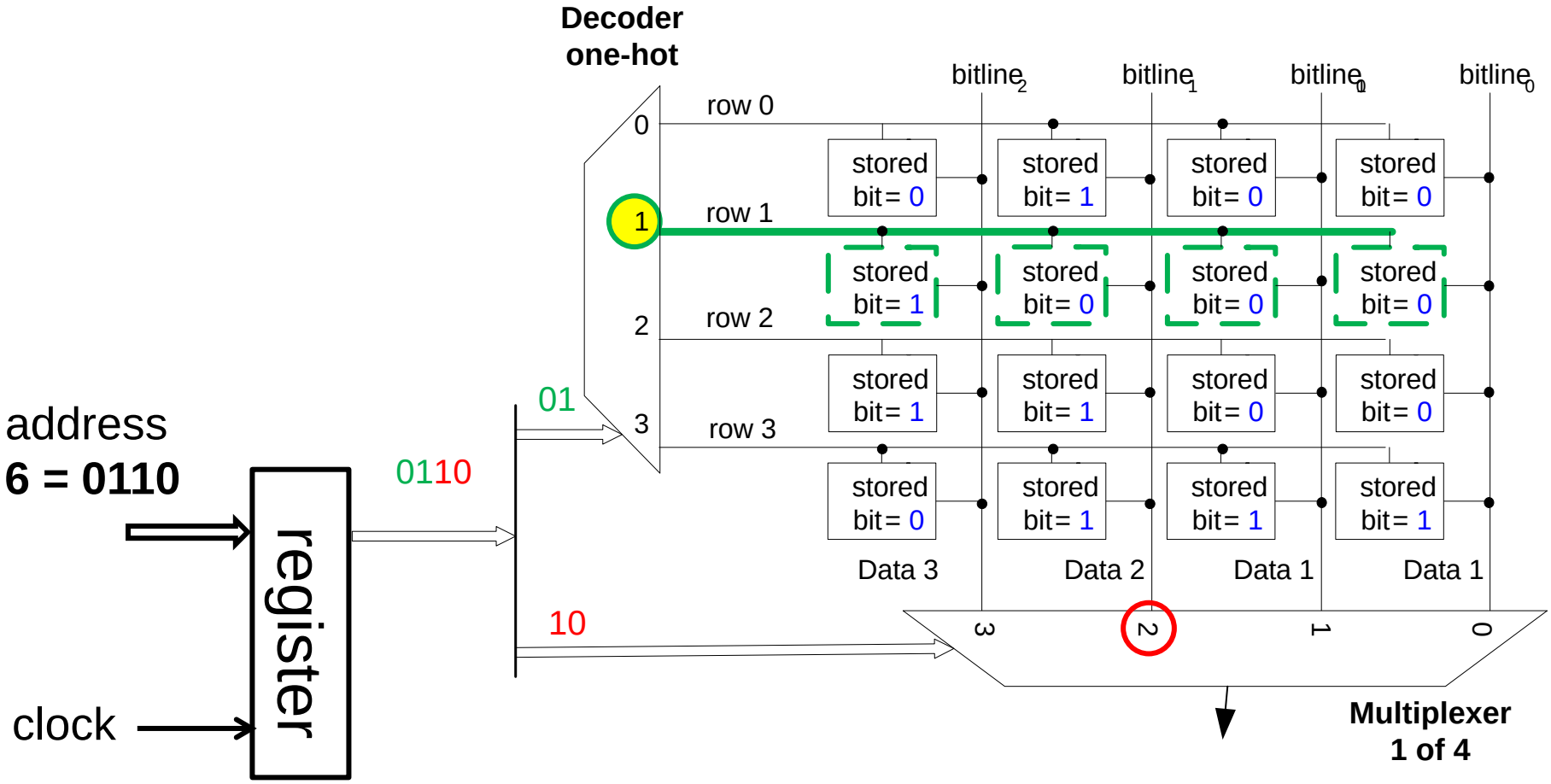
Address value is waiting for the rising edge of clocks

Memory matrix - example 2/4



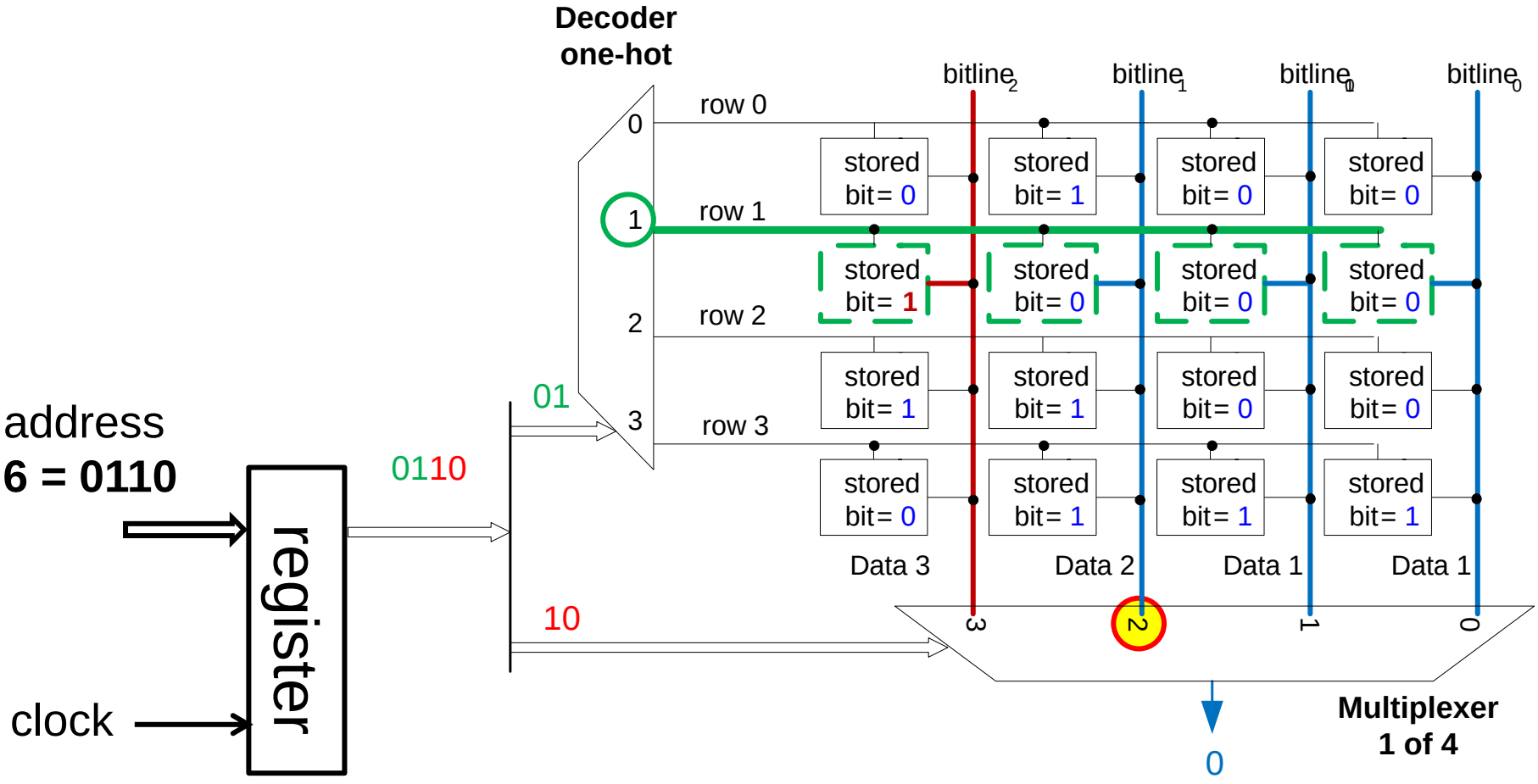
Address was loaded on the rising edge of clocks and divided to two parts, higher and lower bits

Memory matrix - example 3/4



Output of one-hot decoder 1 from N aktivate row and of cell in it.

Memory matrix - example 4/4



The cells are connected to bitlines and the output multiplexer selects one value - Data 2 = 0

Terminologie kolem pamětí

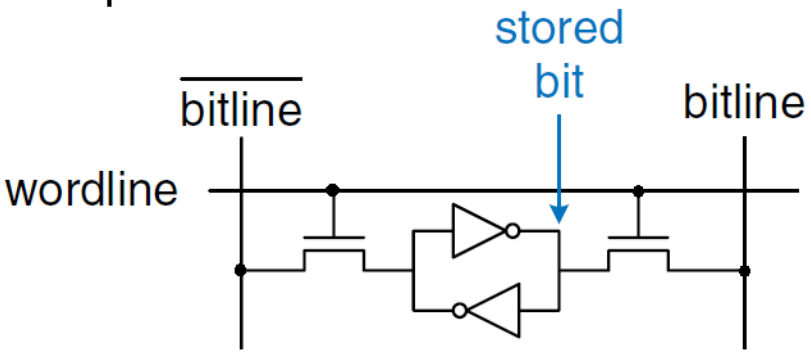
- Typy pamětí RWM (RAM), ROM, FLASH,
- Provedení RAM pamětí:
  SRAM (statická), **DRAM** (dynamická).

- RAM = *Random Access Memory – paměť s **libovolným** přístupem*

typ paměti	počet tranzistorů	plocha na 1 bit	dostupnost dat	latence
SRAM	cca 6	< 0,1 μm^2	vždy	< 1ns – 5ns
DRAM	1	< 0,001 μm^2	potřebuje refresh	desítky ns

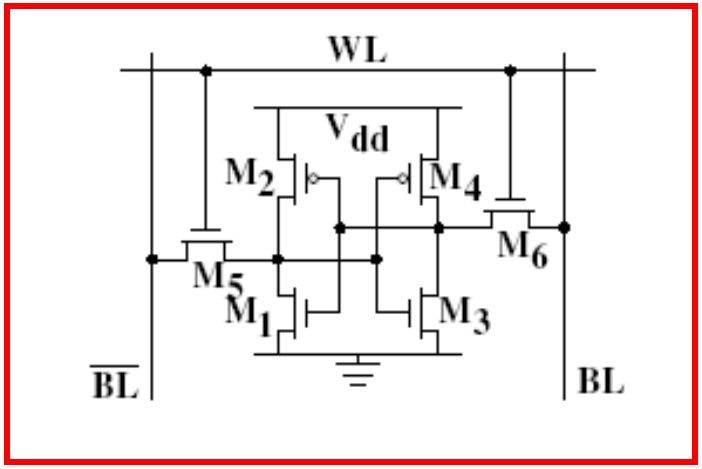
Typický čip a buňka SRAM

Princip:

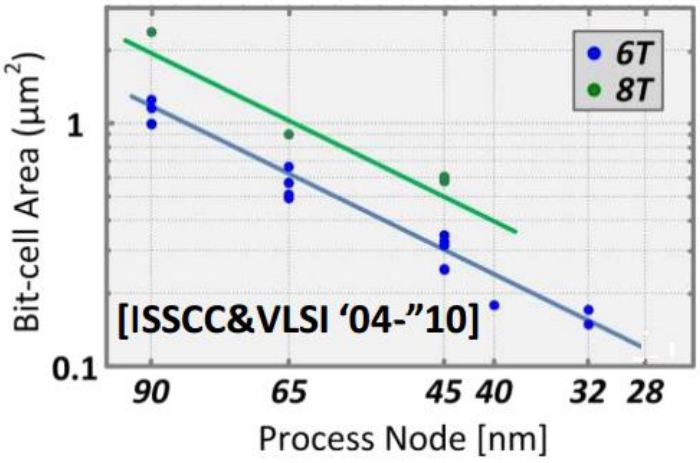


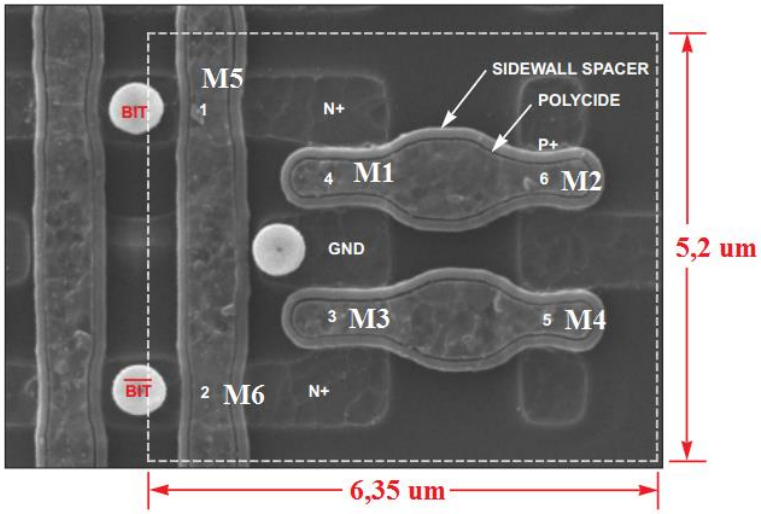
SRAM paměťová buňka

6-transistorů CMOS, existují 4 trans verze



Plocha paměťové buňky:

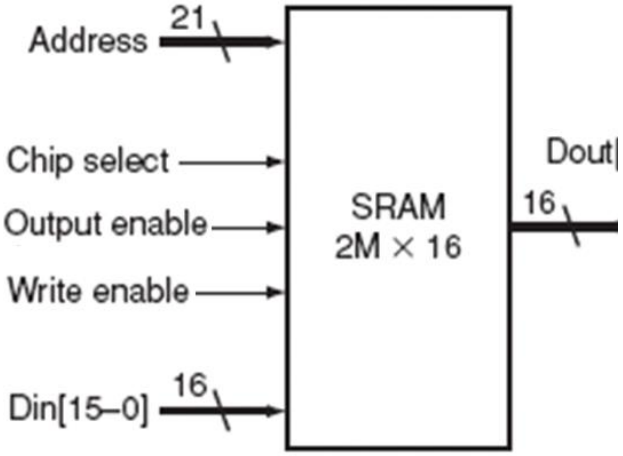




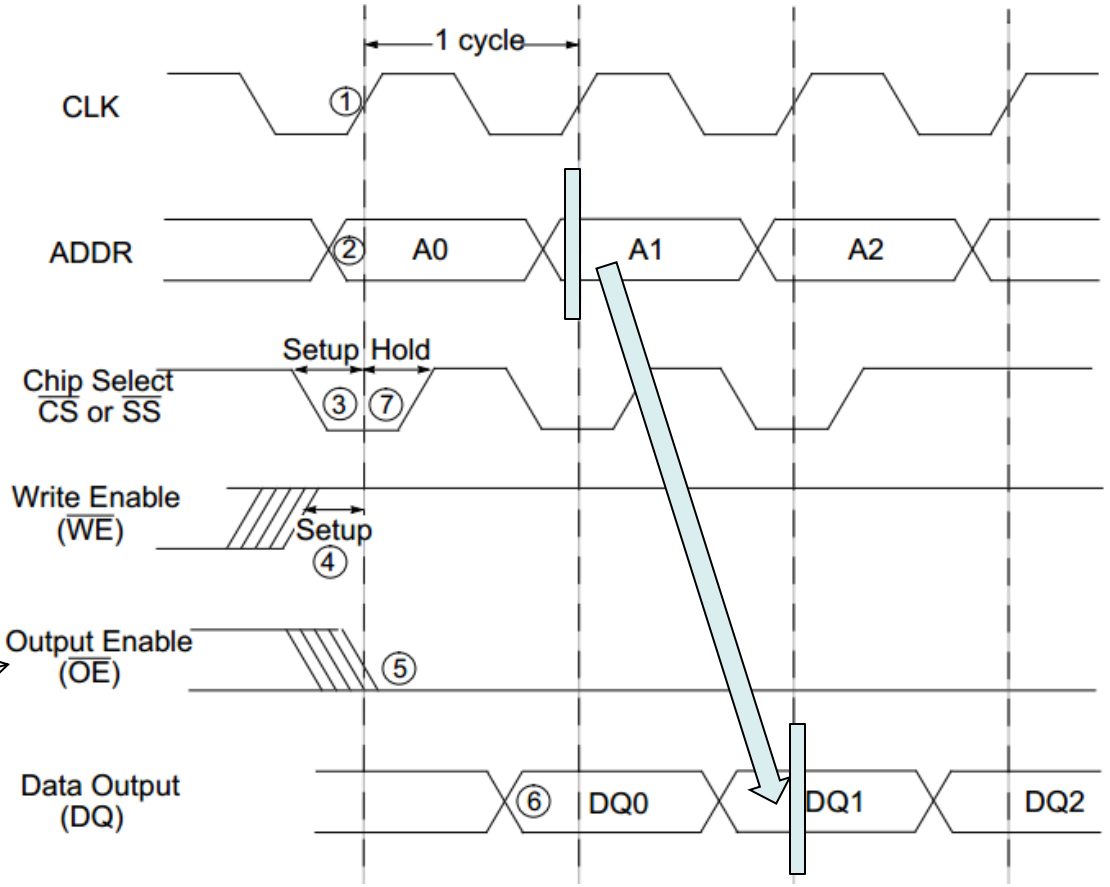
http://educypedia.karadimov.info/library/SEC08.PDF

Typický čip a buňka SRAM

Typický SRAM čip



Příklad čtení – typicky synchronní :

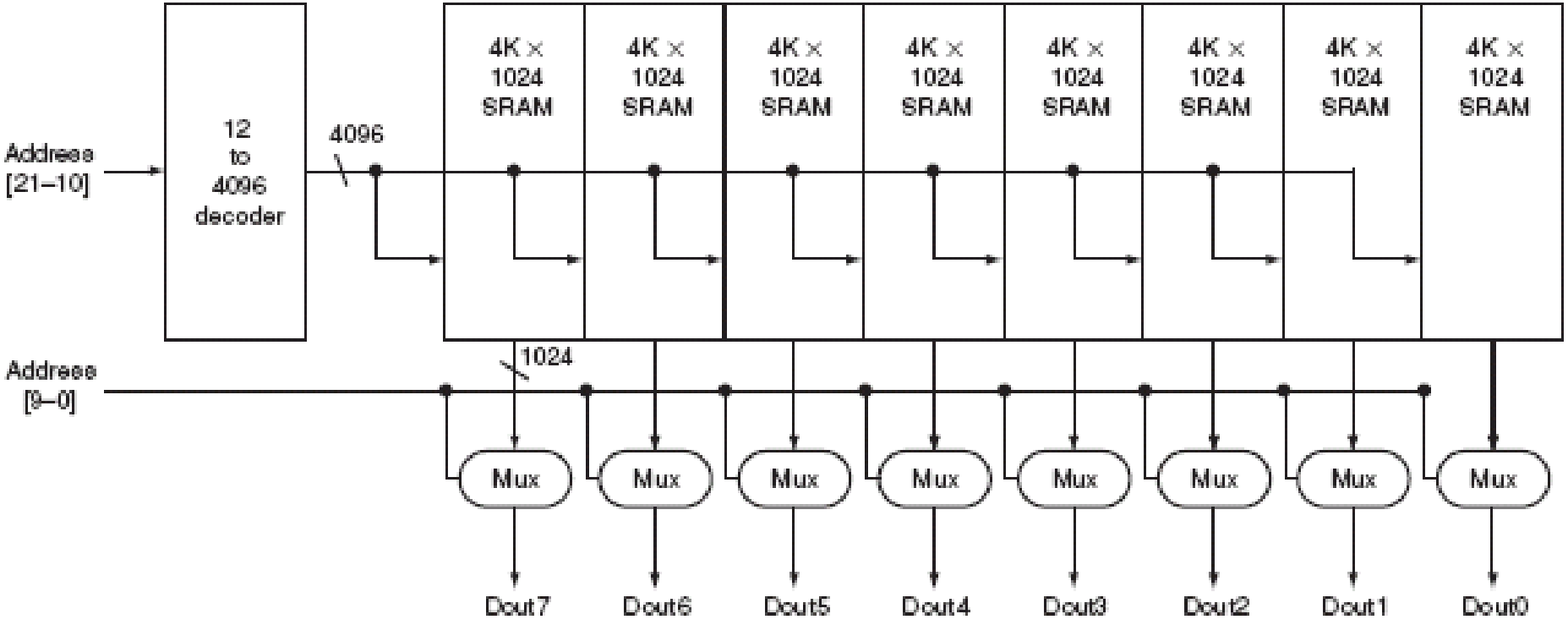


OE signál je asynchronní

https://www.ece.cmu.edu/~ece548/localcpy/sramop.pdf

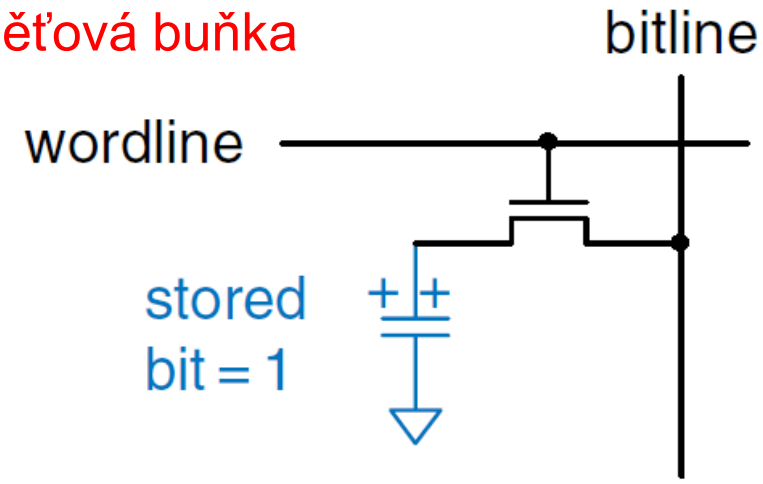
Typický čip a buňka SRAM

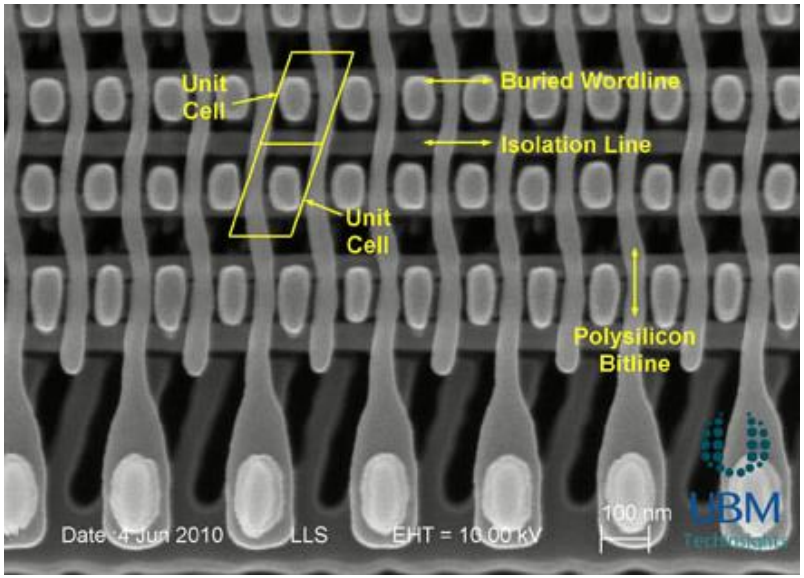
Větší paměť?



Detail paměťové buňky dynamické paměti

Jednotranzistorová dynamická paměťová buňka





nMOS tranzistor představuje přepínač, který připojí (nebo ne) kondenzátor na vodič „bitline“. Připojení je řízeno vodičem „wordline“.

Obrázek: http://www.eetimes.com/document.asp?doc_id=1281315

Capacitor

Commercial DRAM parameters	
	Capacity fF [femtofarad]
Storage capacitor	from 10 fF to 50 fF
Bit line	around 2 fF

[Source: l'INSA de Toulouse]

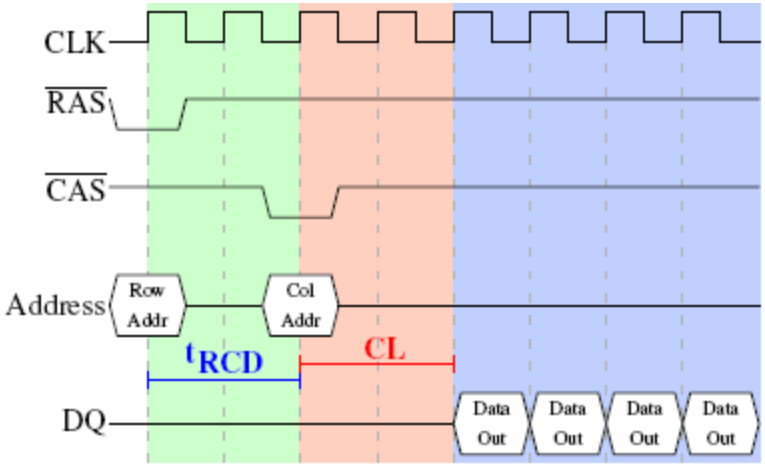
fF - femtofarad
An SI unit of electrical capacitance equal to 10^{-15} farads.

10^{-6} F = 1 μF = 10^{3} nF = 10^{6} pF = 10^{9} fF

~9 fF - the capacitor created by two 1 mm^2 plates
in 1 mm distance inside vacuum

Vývoj DRAM paměťových čipů v čase

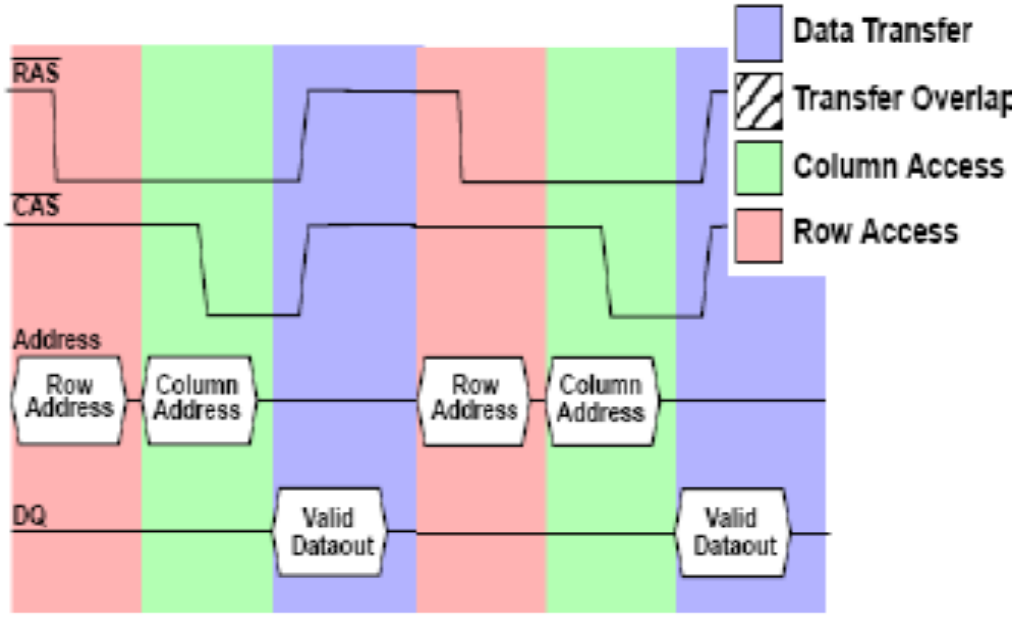
Rok	Kapacita	Cena[$]/GB	Doba přístupu [ns]
1980	64 Kb	1 500 000	250
1983	256 Kb	500 000	185
1985	1 Mb	200 000	135
1989	4 Mb	50 000	110
1992	16 Mb	15 000	90
1996	64 Mb	10 000	60
1998	128 Mb	4 000	60
2000	256 Mb	1 000	55
2004	512 Mb	250	50
2007	1 Gb	50	40



RAS – Row Address Strobe,
CAS – Column Address Strobe

Klasická DRAM – asynchronní rozhraní

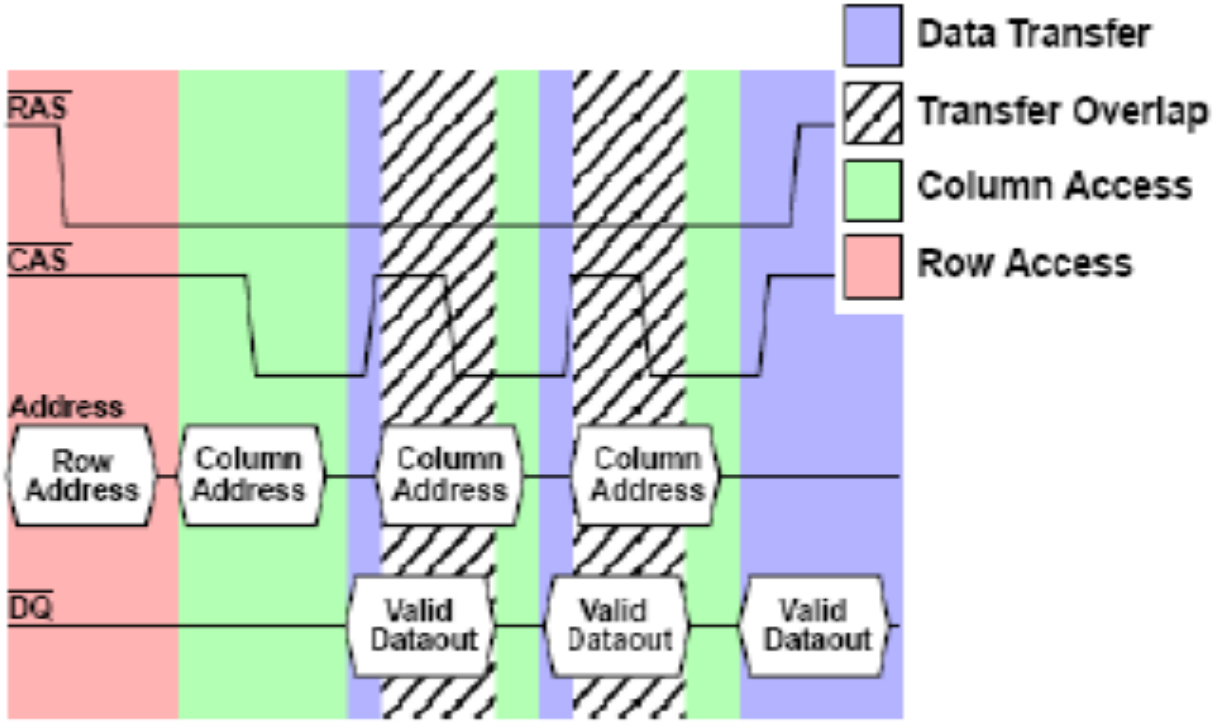
- Důvod rozdělení adresy na 2 části byl dán malým počtem pinů původních DRAM pouzder.
- Toto rozdělení se dodnes zachovává. Sice pouzdro už není problém, ale spojení RAS a CAS by nepřineslo výhodu. Proč?



RAS – Row Address Strobe,
CAS – Column Address Strobe

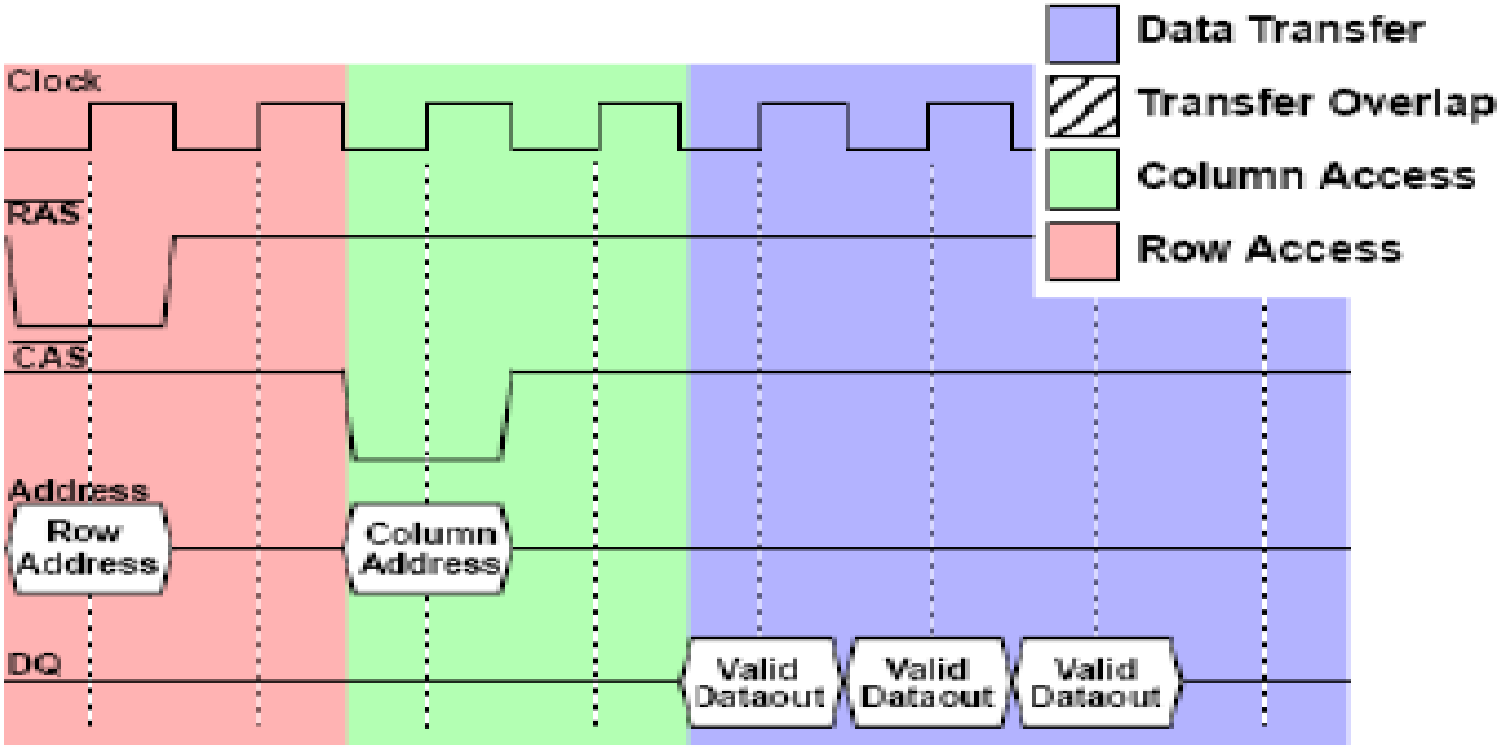
EDO DRAM – cca 1995

- EDO DRAM má registr na výstupu, což umožní překrýt následující CAS s čtením předchozích dat.

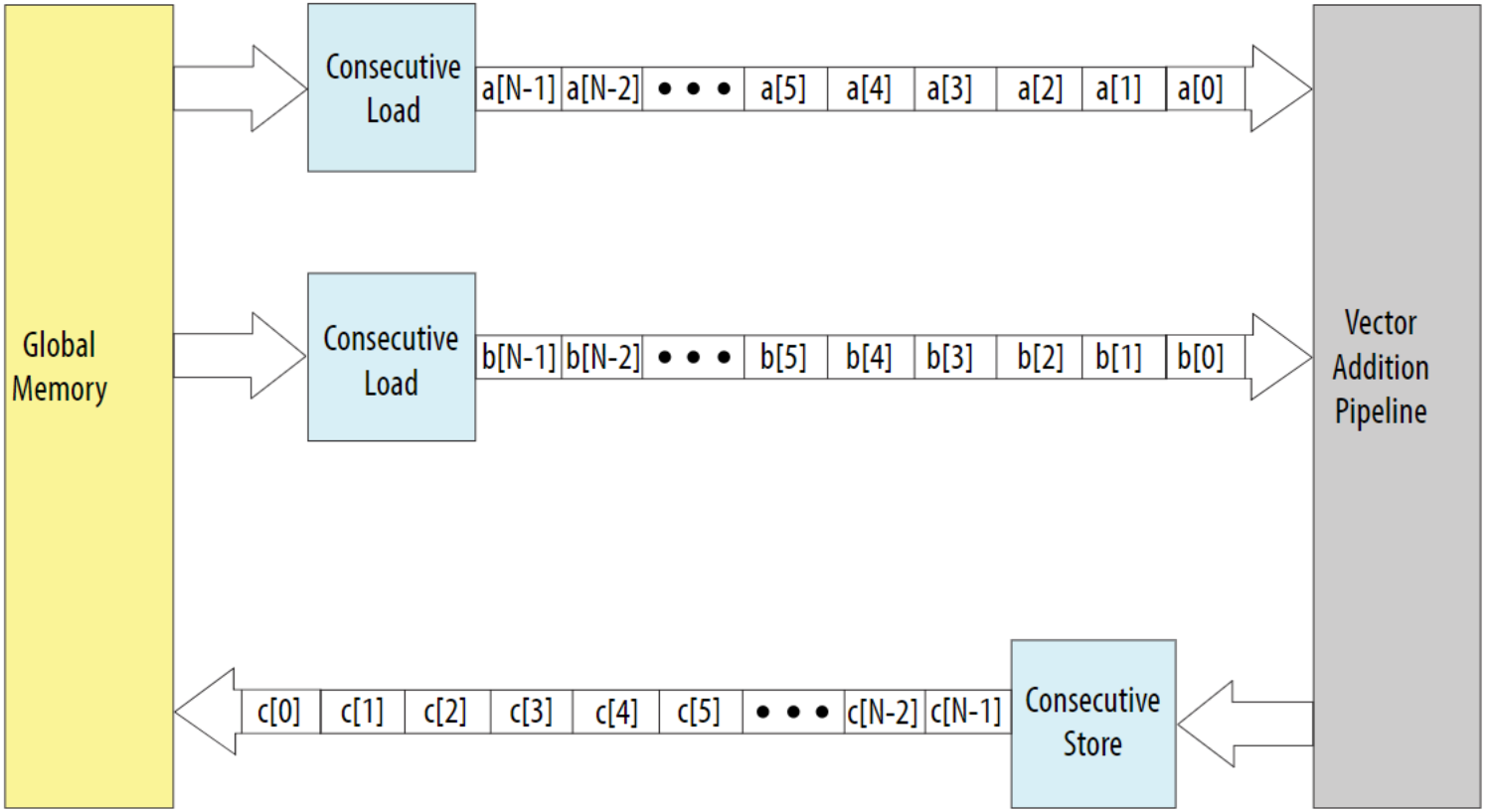


SDRAM – konec 90.let – synchronní DRAM

- SDRAM čip obsahuje čítač, který umožňuje nastavit délku souvislého (burst) čtení/zápisu dat.



Consecutive Read/Write



SDRAM – paměť současnosti

- SDRAM – frekvence až 100 MHz, 2.5V
- DDR (*double data rate*) SDRAM – použití obou hran CLK při přenosu dat, napájení 2.5V, I/O bus clock 100-200 MHz,
  rychlost přenosu 0.2-0.4 GT/s (gigatransfers per second)
- DDR2 SDRAM – snížení spotřeby použitím napětí 1.8V,
  frekvence IO bus 400 MHz, 0.8 GT/s
- DDR3 SDRAM – snížení spotřeby použitím napětí 1.5V, IO bus frekvence až 800 MHz, 1.6 GT/s
- DDR4 SDRAM - napětí 1,05 – 1,2V, I/O bus clock 1.2 GHz, 2.4 GT/s
- DDR5 *SDRAM - očekávají se 2019-20, plánovaná rychlost ~6 GT/s*

Všechny inovace vylepšují jenom propustnost čtení dat,
nikoli latenci čtení první položky.

Další paměti

- QDRx SDRAM (Quad Data Rate) - nejsou však 2* rychlejší, pouze umožňují současné čtení i zápis, jelikož mají oddělené hodiny pro RD a WR, zatímco DDR jsou naopak efektivnější než QDR pro jeden typ přístupu.
- **GDDR** SDRAM - dnes až typ GDDR6, vhodné pro grafické karty
  - založené na DDR pamětech.
  - rychlost se zde akceleruje rozšířenou výstupní sběrnicí.
- Dále ještě existují paměti a moduly RDRAM (RAMBUS DRAM), které ovšem mají zcela odlišné rozhraní i způsob použití. Pro patentní spory se od roku 2003 nepoužívají v osobních počítačích.