Cesta k prvnímu mikroprocesoru

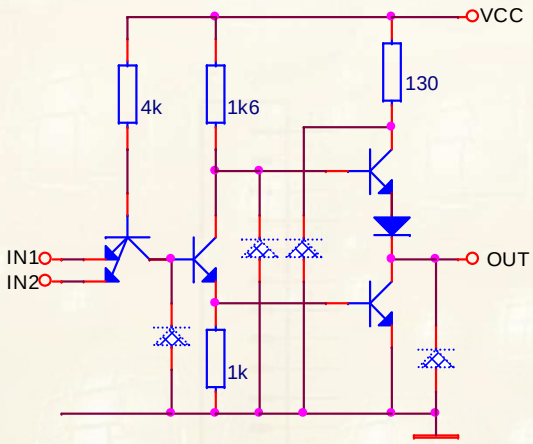
OD LOGICKÝCH OBVODŮ K μP – VLASTNOSTI VÝSTUPŮ LOG.OBVODŮ

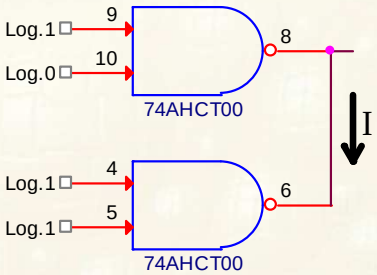
Logické obvody se vyrábí se třemi různými typy výstupů.

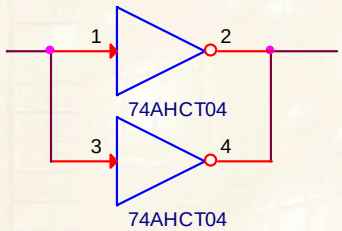
- Obvody s dvoustavovým výstupem
- Obvody s otevřeným kolektorem
- Obvody s třístavovým výstupem

Výstup obvodu s dvoustavovým výstupem

- Stav log.0 (Uvýst= typ. 0,1V až 0,2V, závisí na proudu)
- Stav log.1 (Uvýst= typ. 3,4V bipolární technologii, Ucc=5V, Uvýst= typ. Ucc-0,1V CMOS). Počet připojitelných vstupů je stanoven větvením nebo kapacitou vstupů.

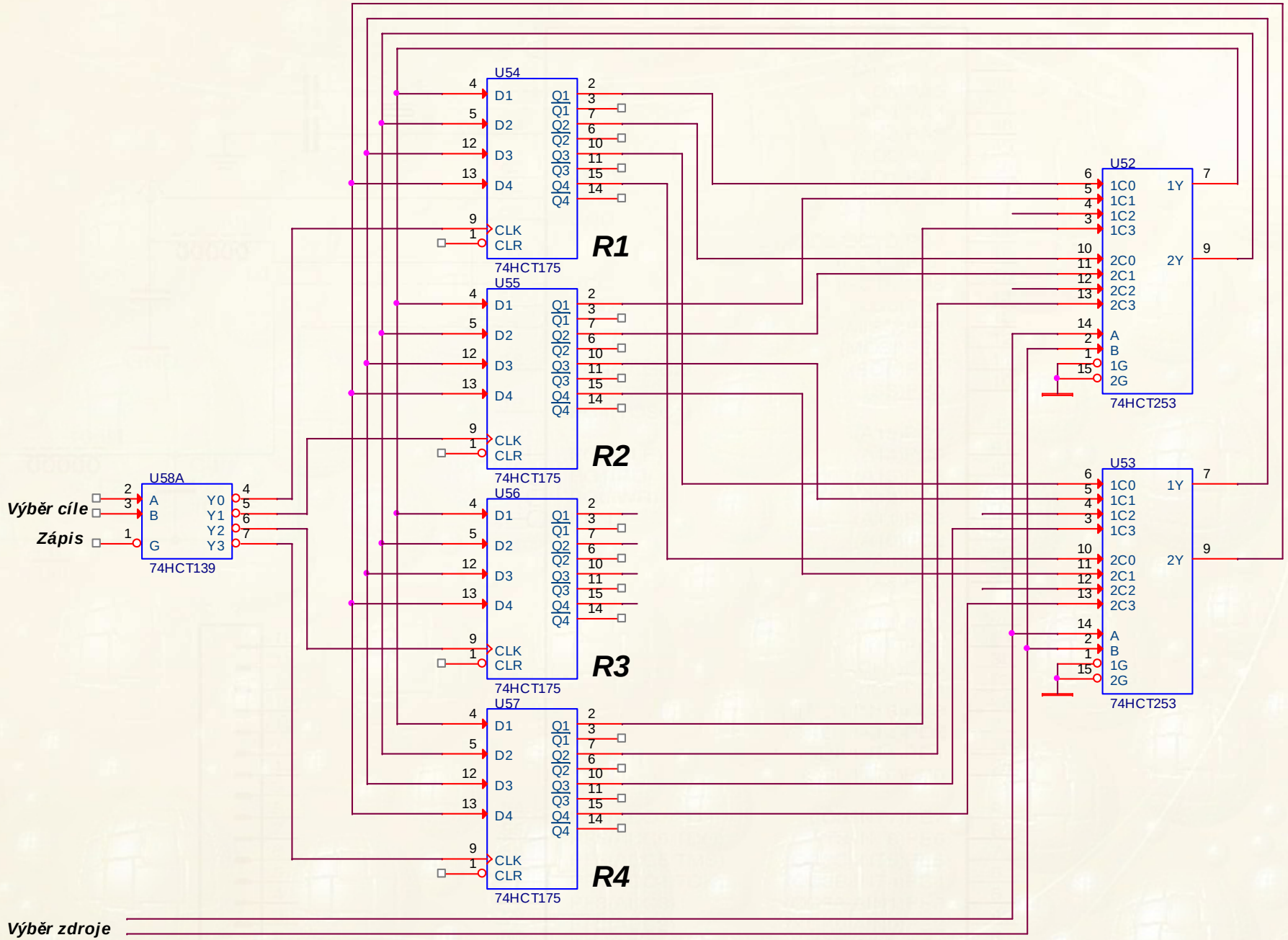






Okolo 1965 se realizovaly složité řadiče, u kterých bylo potřeba zajistit propojení řady registrů. ⇒ realizovatelné **multiplexory**

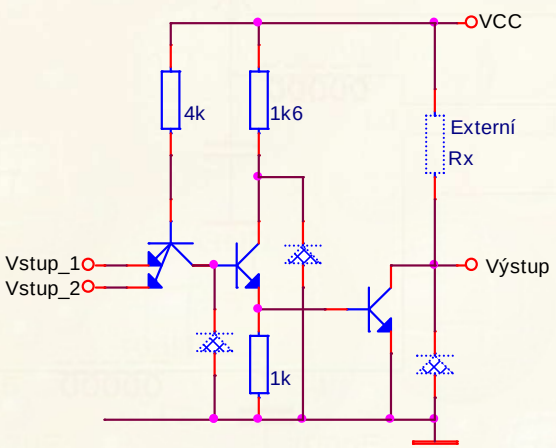
PŘENOS MEZI REGISTRY S DVOUSTAVOVÝM VÝSTUPEM



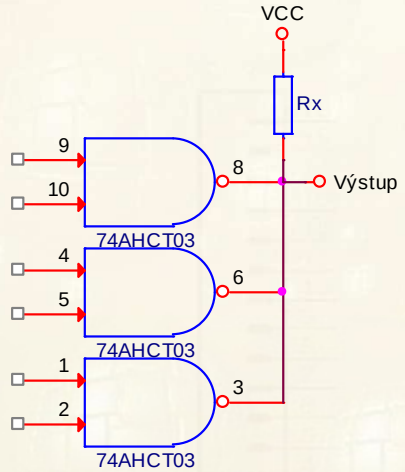
- **Jiná cesta** – **sběrnice** jako soustava propojovacích vodičů k níž jsou připojeny vstupy i výstupy registrů.

  Zkratům mezi výstupy obvodů je zabráněno úpravou

výstupů s otevřeným kolektorem



- Každý vodič sběrnice připojen přes odpor k napájení
  - Velikost odporu výrazně ovlivňuje rychlost přechodu z log.0 do log.1.
  - Problém připojení vstupu rychlého obvodu



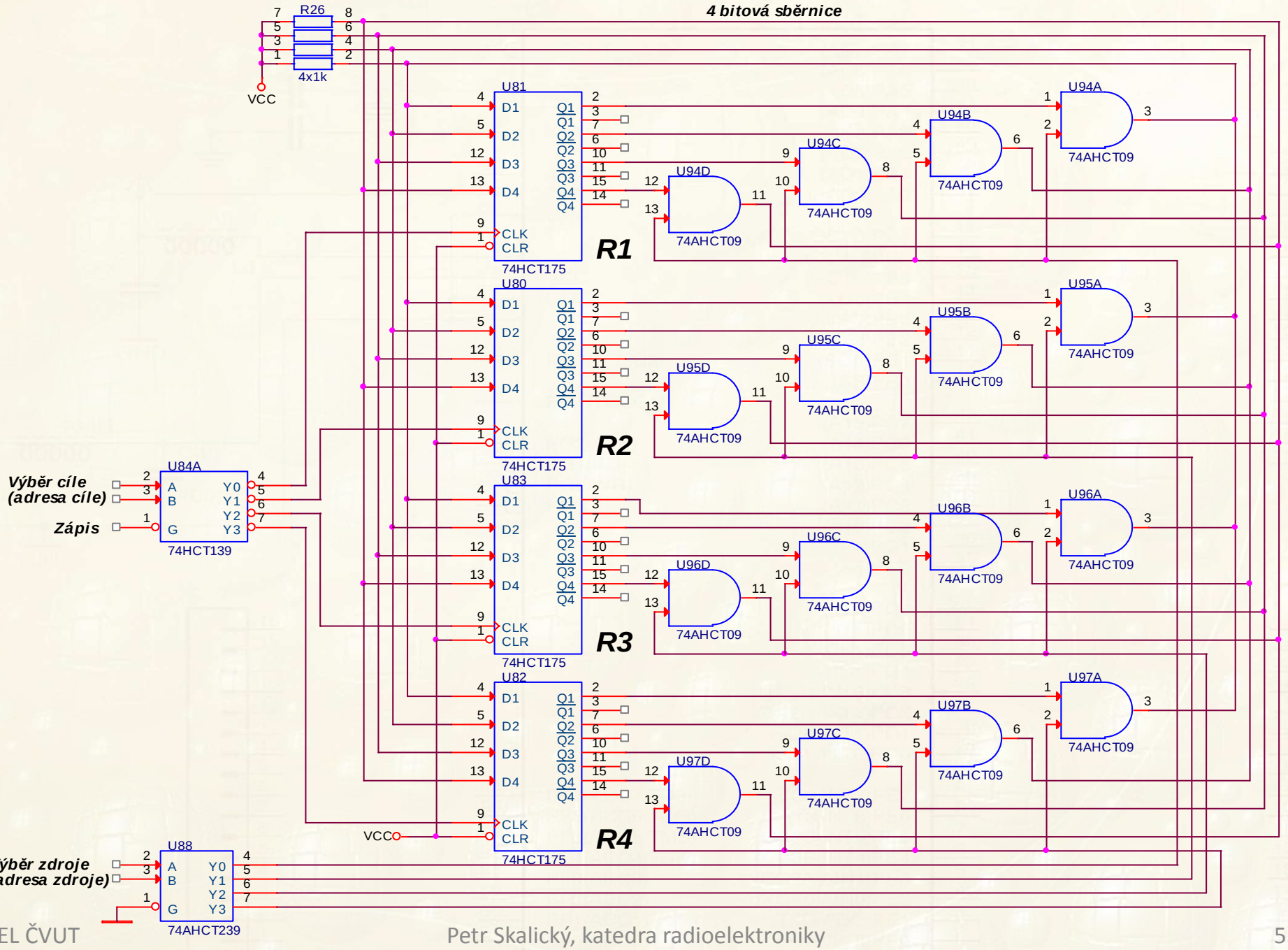
Obvody NAND s otevřeným kolektorem realizují funkci AND-OR-INVERT

$$Y = \overline{A.B}.\overline{C.D}.\overline{E.F} = \overline{\overline{A.B}.\overline{C.D}.\overline{E.F}} = \overline{A.B + C.D + E.F}$$

Označováno jako **montážní součin** nebo **součet**.

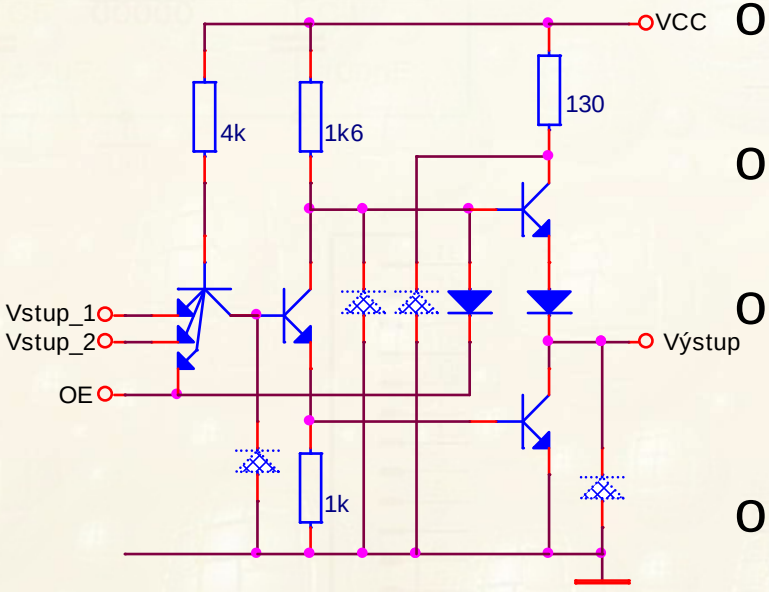
Současné využití - synchronizace různě rychlých obvodů s možností stanovení priority (I^2C), sloučení několika žádostí o přerušení.

PŘENOS MEZI REGISTRY S OTEVŘENÝM KOLEKTOREM



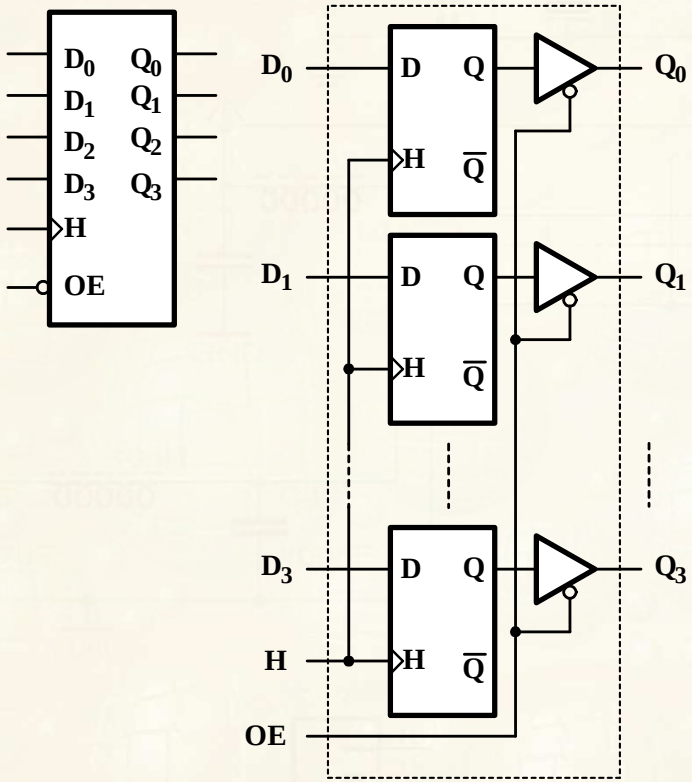
OD LOGICKÝCH OBVODŮ K μP – VLASTNOSTI VÝSTUPŮ LOG.OBVODŮ

- Otevřený kolektor - problémy s časovým zpožděním a výkonovým zatížením výstupů ⇒ nahrazen obvody s **třístavovým** výstupem.
- Dva stavy log.0, log.1 a ve třetím stavu jsou oba koncové tranzistory jsou uzavřené – **stav vysoké impedance (Z).**
- Tento stav ovládá vstup označovaný OE (obvykle log.1 = výstupní vývod ve stavu Z - svodový odpor stovky kΩ až jednotky MΩ, kapacita jednotky pF).

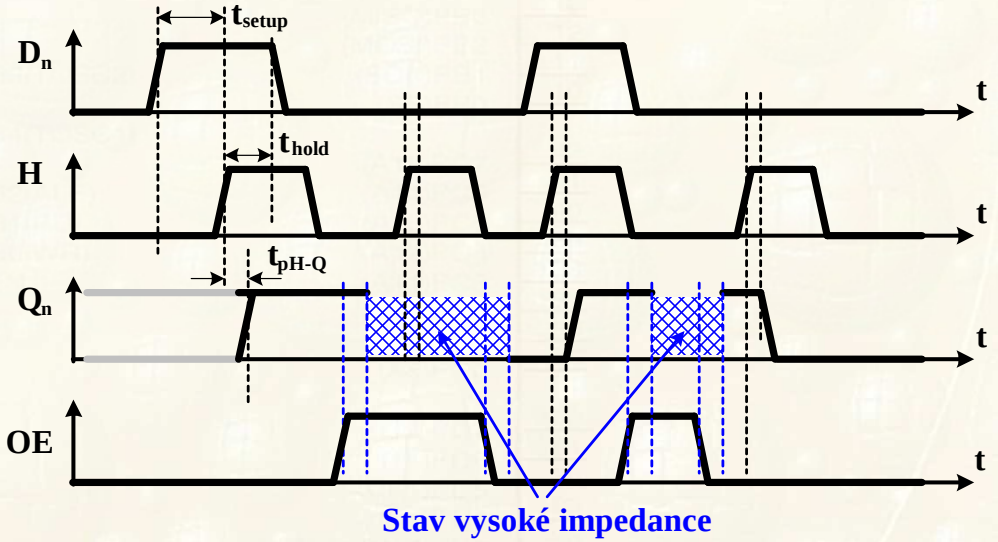


- **Výstupy** obvodů mohou být spojeny
- **Pouze jeden obvod** smí být aktivní (výstup v log.1 nebo log.0)
- Ostatní připojené obvody musí mít **výstup v daném okamžiku** ve stavu Z.
- Připojení vstupů a výstupů k vodiči umožňuje obousměrnou komunikaci vodiče nazýváme **datovou sběrnicí**.
- Zdroj a cíl přenosu určuje tzv. **adresová sběrnice**.

Registr, Latch - s třístavovým výstupem

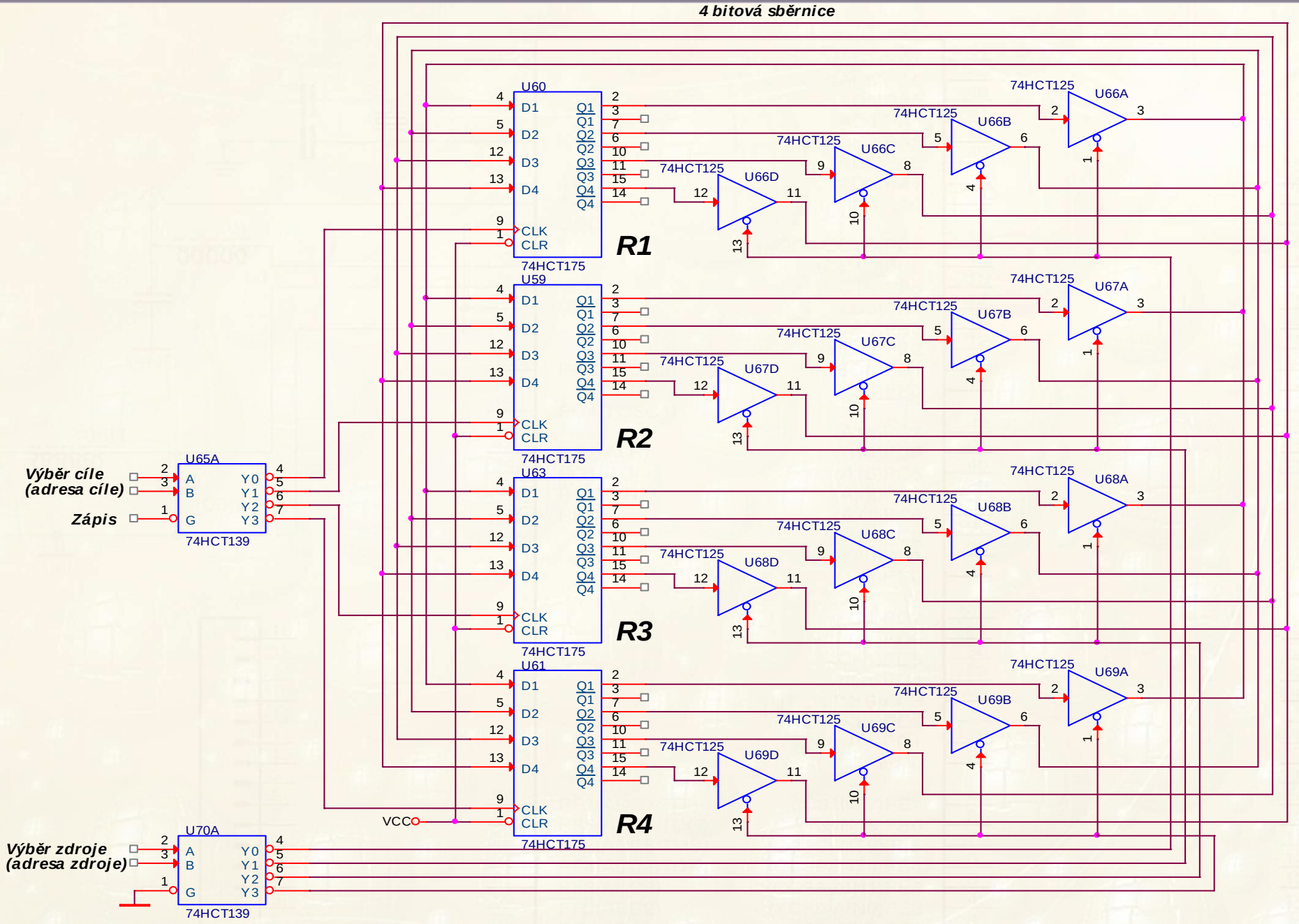


Hodinový signál umožňuje uložit vstupní informaci, která je na výstupech k dispozici pouze, je-li budič aktivován (OE=0).

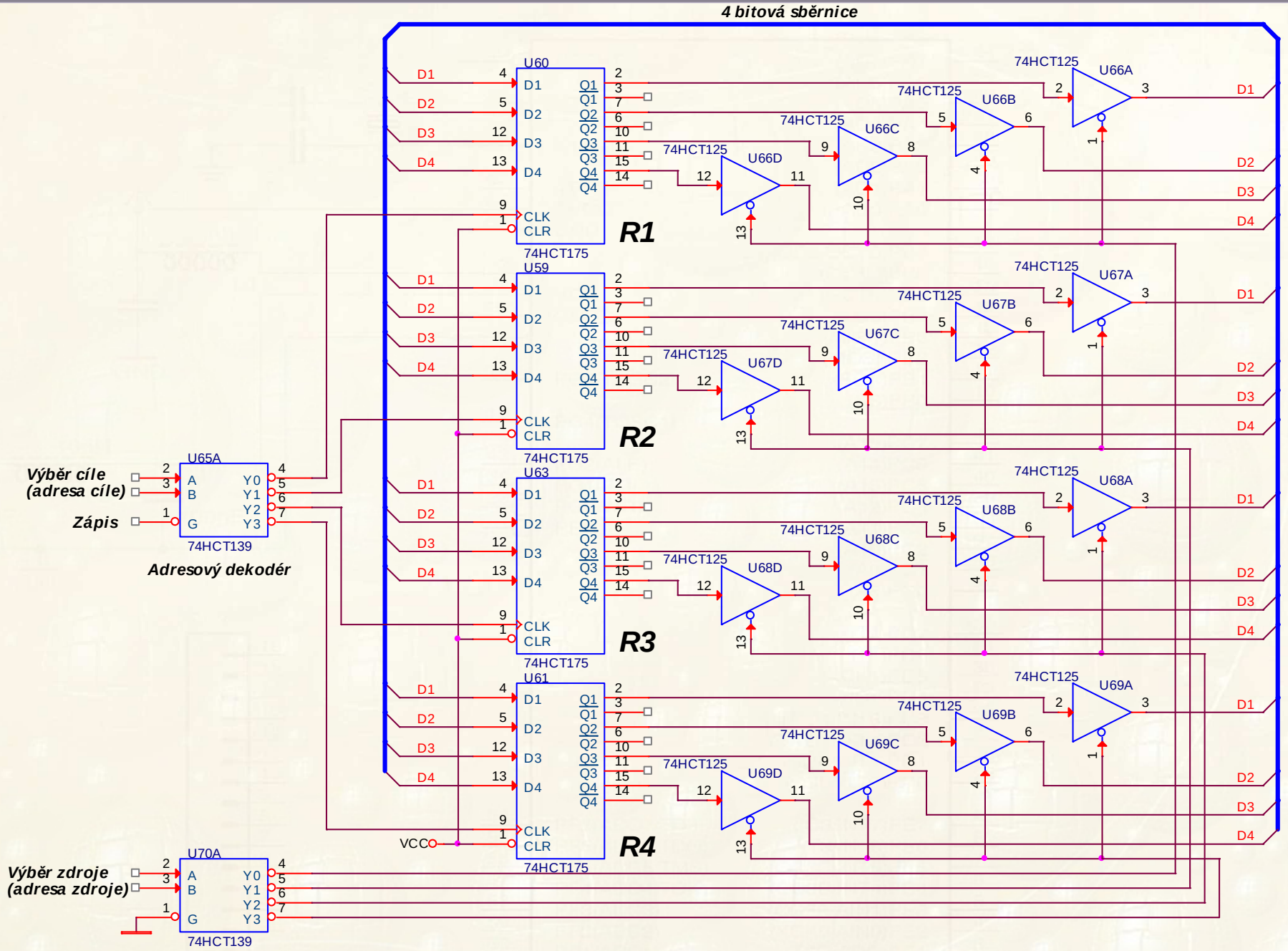


Použití: **Vyrovnávací paměť, výstupní brána µP** (OE=0). **Paměť** vnitřních proměnných LSO (OE=0). **Vstupní brána uP** systému (OE=funkce (adresy a řídícího signálu čtení).

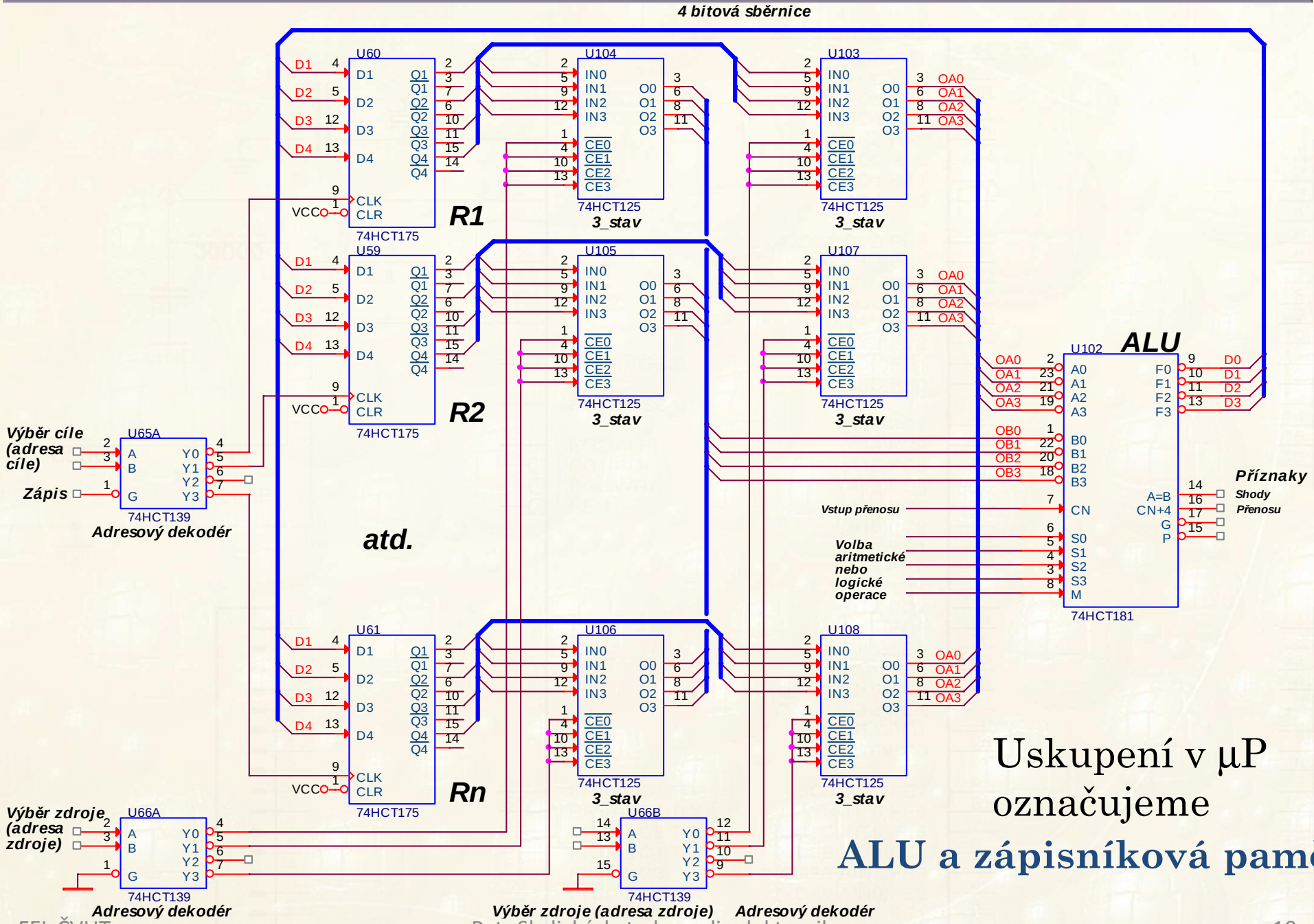
PŘENOS MEZI REGISTRY S TŘÍSTAVOVÝM VÝSTUPEM



PŘENOS MEZI REGISTRY S TŘÍSTAVOVÝM VÝSTUPEM



PROPOJENÍ REGISTRŮ S ARITMETICKO LOGICKOU JEDNOTKOU



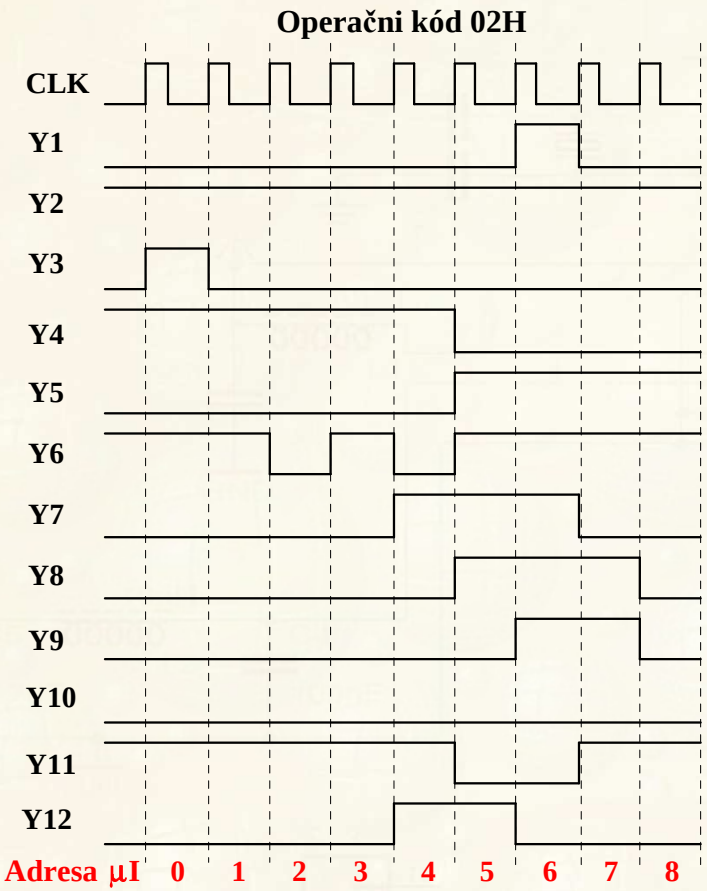
Uskupení v µP označujeme

ALU a zápisníková paměť

ŘÍZENÍ ALU A ZÁPISNÍKOVÉ PAMĚTI

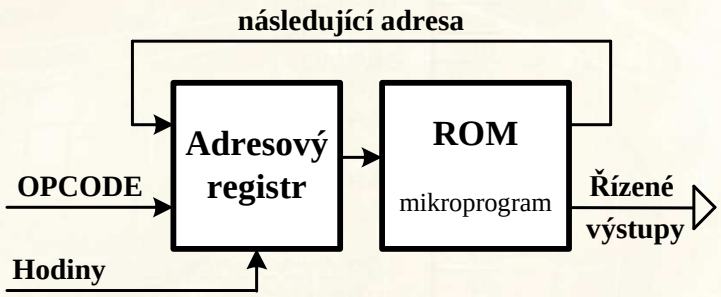
- K vykonání potřebné operace potřebujeme **řadič**, který aktivuje zdroje informace, typ realizované operace a cílový registr (případně paměť). Řadič následně generuje řídicí signály k vykonání požadované operace.
- Realizovanou operaci určuje tzv. **operační kód** doplněný potřebnými informacemi k jejímu vykonání.
- Každá instrukce (operační kód) má pevné uspořádání bitů s potřebnými informacemi.
- Pro realizaci algoritmu (programu) musíme mít nástroj k postupnému čtení jednotlivých instrukcí **(Fetch)**.
  - Fetch je součástí instrukce. Pro velký počet instrukcí (operací) dochází k **degradaci řadiče**.
  - Fetch bude realizován podprogramem ⇒ **Efektivnější řešení**

MIKROPROGRAMOVATELNÝ ŘADIČ



Adresa	Obsah ROM	Adresa	Obsah ROM
020h	42E1h	025h	8F26h
021h	42A2h	026h	1F37h
022h	40A3h	027h	5B28h
023h	42A4h	028h	4329h
024h	C4A5h		

Adresa mikroinstrukce
Adresa následující mikroinstrukce



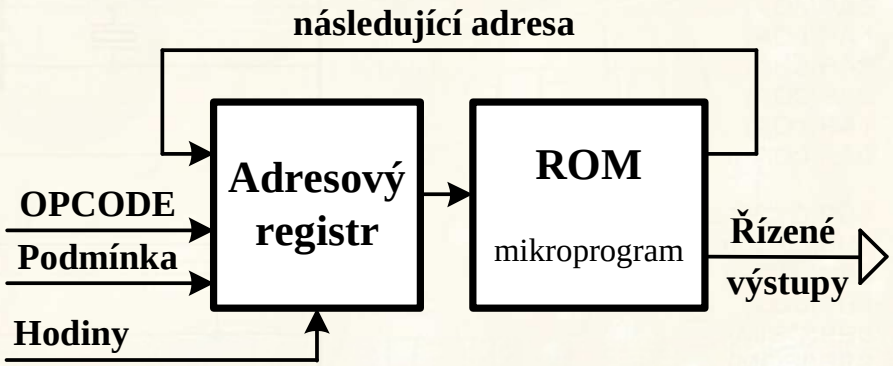
Adresa ROM = Operační kód + následující adresa.

Výstup ROM = **Generovaná sekvence** (mikroinstrukce) + následující adresa mikroprogramu.

Pro obrázek vlevo:

- Výstupy ROM = Q15÷Q4 signály Y12÷Y1, Q3÷Q0 následující adresa,.
- Adresa ROM = An÷A4 operační kód (zde 02h), A3÷A0 následující adresa.

Konfigurace z obrázku umožňuje:



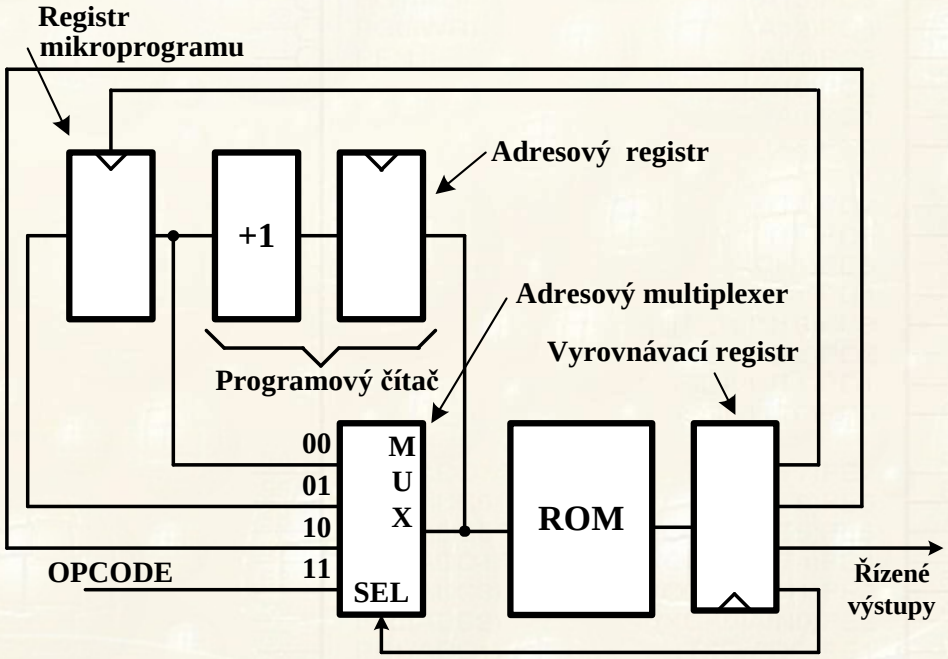
❖ **Sekvenční čtení ⇒ řídící signály** pro realizaci operačního kódu, **sekvenci pro načtení dalšího operačního kódu (Fetch)**.

❖ **Realizovat skok**

- Umožňuje – **skok do podprogramu**
- Neumožňuje – **návrat z podprogramu**
- Omezeně umožňuje – **podmíněné větvení**

Modifikace struktury

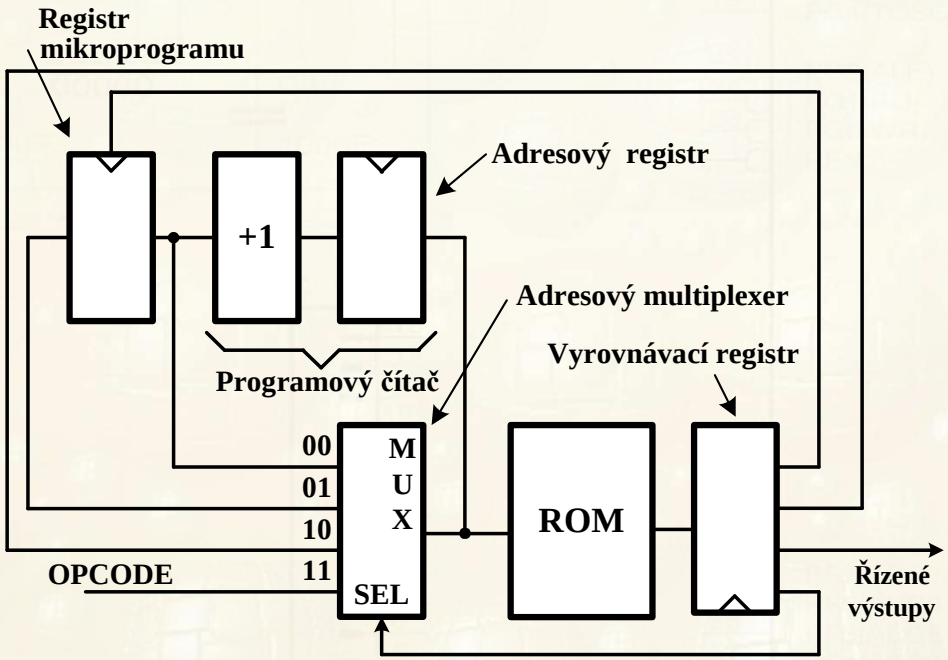
- **Programový čítač** (Program counter) → adresový registr doplněn o inkrementující sčítačku.
- **Registr** mikroprogramu → **jednoúrovňový zásobník** pro uchování **návratové adresy** (návrat z mikropodprogramu).
- **Multiplexer** – výběr následující adresy.
- **Vyrovnávací registr** – synchronizace všech signálů a eliminace zpoždění způsobeného dobou vybavení z ROM.



Adresa na výstupu multiplexoru

- **SEL=00 - Programový čítač** (sekvenční zpracování µprogramu)
- **SEL=01 - Registru mikroprogramu** (návrat z podprogramu)
- **SEL=10 - Vyrovnávací registr** (skok, volání podprogramu)
- **SEL=11 - Nový operační kód**

Jednotka umožňuje



- **Sekvenční čtení**
- **Skok na libovolnou adresu**
- **Zavolání podprogramu** (vždy jen jednoho)
- **Návrat z podprogramu.**

Změnou ovládání MUX lze zavést **podmíněné větvení**. Případně je možné přidat do řadiče reakci na vstupní signály - **stavové příznaky**.

VZNIK PRVNÍHO MIKROPROCESORU

Stolní kalkulačky - výpočetní jádro = propojení registrů s ALU (74181) spolu s řídící jednotkou. Operace realizovány

- Paralelně s Nx4 bity ALU kaskádně řazenými
- Sériově s využitím jedné 4-bitové ALU

⇒ Obě řešení potřebovala řadu podpůrných obvodů a **jednoúčelový řadič zajišťující všechny** výpočty (**funkce**) kalkulačky.

Myšlenka – vytvořme instrukce (operace ALU), z kterých složíme **požadovaný výpočet. Instrukce uložené v programové paměti budou postupně čteny a vykonávány. Řadič bude jednoúčelový pouze s ohledem na realizaci jednotlivých instrukcí nikoliv k funkci celého zařízení.**

Řadič se zjednodušil na generování signálů pro čtení instrukce, dekódování, načtení operandů, vykonání instrukce a uložení výsledku. **Je nezávislý** vůči funkci navrhovaného zařízení.

15.12.1971 byl předveden firmou **Intel** obvod **I4004**, který vznikl v rámci vývoje kalkulátoru pro firmu Busicom 141-FP.

VÝVOJ MIKROPROCESORŮ

- 1971 – (15.prosince) **I4004** první procesor (4-bitový) navržen pro snazší realizaci stolní kalkulačky. CPU se skládalo z 4 integrovaných obvodů a obsahovalo cca 8000 tranzistorů. Příliš neuspěl stejně jako I8008 a Intel zpětně odkoupil realizovaný vývoj .
- 1974 uveden legendární procesor **I8080**, který dosáhl masového využití nejen ve vojenských a průmyslových aplikacích, tak i prvních osobních počítačích. CPU se skládala ze 3 obvodů a celý procesorový systém bylo možné realizovat v 8 a více integrovaných obvodech.
- 1976 první jednočipový procesor 8048 předchůdce řady **8051** (1980).
- 1979 uveden první **signálový procesor I2920**. Naznačil nový možný směr vývoje procesorů. Jeho následovníci již mají výrazně odlišnou architekturu (modifikovaná Harvardská struktura, dvě sběrnice, jednotka MAC).
- 1982 - procesor **80286**, první specializovaný procesor pro osobní počítače.
- Koncem 70 let vznikají první vývojové systémy, **překladače assembleru**, **aritmetické knihovny** s pohyblivou čárkou a **operační systémy CP/M** (1982 předchůdce systému **DOS**)
- 2001 – **NIOS** (Altera) první komerční implementace procesoru v programovatelném poli **LCA** (FPGA)

Mikroprocesor

Mikroprocesor

- **sekvenční obvod s pevně nastaveným algoritmem**
- **postupně čte jednu nebo více instrukcí uživatelského programu**
- **přečtené instrukce zpracovává postupně nebo současně (paralelně)**
- Jako každý LSO, musí být po připojení k napájení uveden do **počátečního stavu** ⇒ algoritmus musí začít z definovaného stavu.
- Počáteční stav je dán **čítačem instrukcí** (obvykle PC=0000) a stavem **důležitých registrů** (přerušovací systém, interní periferie, V/V obvody, atd.) před zahájením programu.
- PC (není-li sčítán s kódovým segmentem) obsahuje **adresu**, z které se **bude číst první instrukce programu**.

ALGORITMUS PROCESORU

- **Řadič** zajistí přenesení adresy z PC na **adresovou sběrnici** a vygeneruje řídící signál pro čtení. (Von Neuman z operační paměti), (harvardská struktura z programové paměti)
- Po vybavení hodnoty z paměti a zpoždění na datové sběrnici se objeví **operační kód** první instrukce realizovaného programu, který se uloží do **instrukčního registru** a analyzuje v dekodéru instrukcí.
- Je-li instrukce jednobytová nebo jednoslovná, jsou řadičem generovány řídící signály potřebné k jejímu vykonání.
- Jsou-li k operačnímu kódu potřeba načíst **data** nebo **adresu** je z následující adresy (PC=PC+1) přečten další byte nebo slovo.
- **Nejsou-li všechny části instrukce přečteny, pokračuje procesor ve zvětšování PC v jejich čtení.**

ALGORITMUS PROCESORU

- Je-li k dispozici celá instrukce - **řadič** generuje signály potřebné k **jejímu vykonání**.
- Čítač instrukcí je inkrementován a z **následující adresy je přečtena hodnota, kterou procesor bude chápat jako operační kód.** (*Vyjma případu, kdy procesor je schopen zjistit, že přečtená hodnota není operačním kódem a zareagovat na tuto situaci*).
- **Podle uvedeného algoritmu procesor postupuje od počáteční adresy, nezávisle na tom, zda čtená data jsou smysluplná nebo ne, zda jsou vůbec čtena z nějaké paměti.**

⇒

Uživatelský program (aplikace, operační systém, atd.) musí představovat nekonečnou smyčku.

Program **musí** být ukončen instrukcí skoku nebo instrukcí, která je součástí nekonečného cyklu.

♣ Špatně ukončený program

- Může **ošetřit** operační nebo vývojový systém chybovým hlášením a návratem do systému.
- V aplikaci bez OP nebo VS pokračuje procesor podle **svého algoritmu**.

  Jeho chování závisí na mnoha okolnostech.
- Může proběhnout celou paměť a vrátit se na počáteční adresu.
- Může přepsat část programové (je-li to možné), datové paměti, EEPROM, změnit konfiguraci bran.
- Může vykonávat nechtěnou **nekonečnou smyčku.** Skok na byte (slovo), který z hlediska programu není OP. ⇒ **Dochází k jiné interpretaci** obsahu paměti. **Disinterpretace programu** může být způsobena i dalšími vlivy (napájením, rušením, EMC, špatným návrhem PCB, nevhodnými obvody, atd.).

REGISTRY S ALU

- **Registry (zápisníková paměť, banky registrů)** - soubor obecných a speciálních registrů s rychlým přístupem k ALU.
  - Slouží k uložení proměnných (operandů), mezivýsledků
  - Mohou představovat ukazatel
  - Realizují funkci určenou výrobcem.
  - V jazyce C slouží k vykonání operací
    - Trvale se v nich **neuchovávají** proměnné.
    - Je vhodné je maximálně využít pro operace v jazyce C
      ⇒ Operace v jazyce C je **nevhodné rozdělovat**
  - Registry mohou být přístupné:
    - **Implicitně** - instrukce obsahuje informaci o použitém registru (přímo nebo v operačním kódu)
    - **Paměťově** – v instrukci je obsažena adresa registru nebo odkaz na jejího ukazatele.
    - **Oběma způsoby**

♣ Registry (zápisníková paměť, banky registrů)

- Staré procesory disponují malým počtem registrů (obvykle 8) a střadačem (registr A). Mohou být dostupné pouze implicitně (MOV A,B) nebo jsou i paměťově mapované (MOV B, ACC tj. MOV B, adresa A). Některé mohou zastávat funkci ukazatele do paměti.

- Nové procesory mívají obvykle 32 registrů a střadačem může být každý. Některé mohou zastávat i další funkce např. ukazatele do paměti.

- Procesor ARM disponuje 13 všeobecně použitelnými registry (R0 až R12). Na rozdíl o běžných procesorů má **přímo dostupný čítač instrukcí PC (R15).**

INSTRUKCE

- **Instrukce** – obsahuje operační kód operace doplněný dalšími potřebnými informacemi k jejímu vykonání
  - Registry
  - Konstanty
  - Adresy
  - Ukazatele na adresu

  Formát instrukce (rozložení jednotlivých bitů) **není nutné běžně znát**. Vyjma chyby v překladači nebo realizaci zpětného překladače.

  Čtení instrukce ovlivňuje nejenom počet bitů instrukce, ale také:

  - Šířka programové (instrukční) sběrnice
  - Schopnost procesoru číst z jednoho nebo více paměťových míst najednou

INSTRUKCE

Procesor	Obsah instrukce		
8-bitový	Operační kód (se vším potřebným)		
	Operační kód	Data, Relativní adresa	
	Operační kód	Část Adresy, 16 bit.dat	Část Adresy, 16 bit.dat
16-bitový	Operační kód (se vším), případně za cenu menšího adresového prostoru		
	Operační kód	16 –bitová adresa, data	
32-bitový	Operační kód (se vším potřebným)		
	Dva 16-bitové operační kódy		
?	Operační kód + 32 bitová konstanta nebo adresa		

```
0x08000130   4828    LDR   r0,[pc,#160]   ;@0x0800 01D4
```

32 bitová hodnota v tabulce. Adresa tabulky = instrukce na adrese 0x0800 0130, čteny 4 byty, PC=0x0800 0134, posun 160 = 0xA0 (hexadecimálně)

0x0800 0134 + 0xA0 = 0x0800 01D4

Ve výpisu se objeví **@0x0800 01D4,** znak @ signalizuje, že adresa byla získána relativně.

INSTRUKCE

- **Nevyužití všech 32 bitů** v instrukci ARM ⇒ neefektivní využití paměťového prostoru.
- **Vznik 16-bitových instrukcí** označovaných **Thumb** ⇒ lepší využití prostoru a zrychlení čtení.
- Sada Thumb doplněna o některé 32 bitové instrukce ⇒ sada **Thumb-2** (strojový kód není shodný ARM).
- Rozlišení **instrukcí ARM** od **Thumb** a **Thumb-2.** Adresy na instrukce jsou od ARM odlišeny bitem **b0=1.**
- Směs 16 a 32 bitových instrukcí u JSA vede k nutnosti zarovnávat adresy na modulo 4 (align).
- Stejná situace je u DSP při kruhovém adresování.

♣ Aritmeticko-logická jednotka (ALU)

- Realizuje aritmetické a logické operace
- Nastavuje příznakové bity a jejich stavem je ovlivňována
- Minimum instrukcí pro celou aritmetiku:
  - **Aritmetické sčítání**
  - **Jednotkový doplněk (inverze bitů)**
  - **Logický součin (identifikace hodnoty bitu)**
  - **Podmíněné větvení na základě hodnoty nebo bitu**
- **Má-li ALU větší počet bitů**, než operandy do ní vstupující ⇒ **musí být známo, zda je operand se znaménkem nebo bez znaménka** (informace se nastavuje nějakým bitem).

- **Programový čítač –** obsahuje adresu, z které se bude číst instrukce (případně data/adresa). Obvykle není přímo přístupný.
- **Instrukční registr** – interní registr pro dočasné uchování instrukce pro dekodér instrukcí a řadič procesoru.
- **Řadič - generuje signály** k vykonání instrukce. Činnost řadiče můžeme rozdělit do těchto fází:
  - **Načtení instrukce - (Fetch)**
  - **Dekódování instrukce** - a inkrementace PC. Dekódování určuje o dalším čtení z programové paměti nebo o zahájení vykonání instrukce.
  - **Příprava operandů** – přenos operandů k ALU
  - **Vykonání instrukce** – a zápis výsledku

Řadič může být realizován:

- **Sekvenčním obvodem** s neměnnou logikou. Obvykle používaný u procesorů RISC (tzn. signálové a některé jednočipové procesory).
- **Programový** (mikroprogramovatelný) s možností měnit logiku chování a efektivně realizovat obsluhu mnoha instrukcí za pomoci **mikrokódu.** Využívá se u procesorů CISC s různou dobou trvání instrukcí.

- **Ukazatel zásobníkové paměti SP** – ukazuje do vnitřní nebo vnější datové paměti a slouží:
  - Pro zápis/čtení **návratových adres** při volání/návratu z podprogramů nebo přerušení.
  - Pro dočasné **bez adresné ukládání mezivýsledků** a **proměnných**
  - Pro předávání mnoha proměnných do podprogramu.

Obecně ukazatel zásobníkové paměti ukazuje na:

- Neobsazenou nebo obsazenou adresu v datové paměti
- Při ukládání se SP dekrementuje nebo inkrementuje.

U ARM, je tvořen registrem **R13,** ukazuje na **obsazenou adresu** a při plnění se **dekrementuje** (adresa pro uložení hodnoty se zmenšuje).

Umístění proměnných a zásobníku?

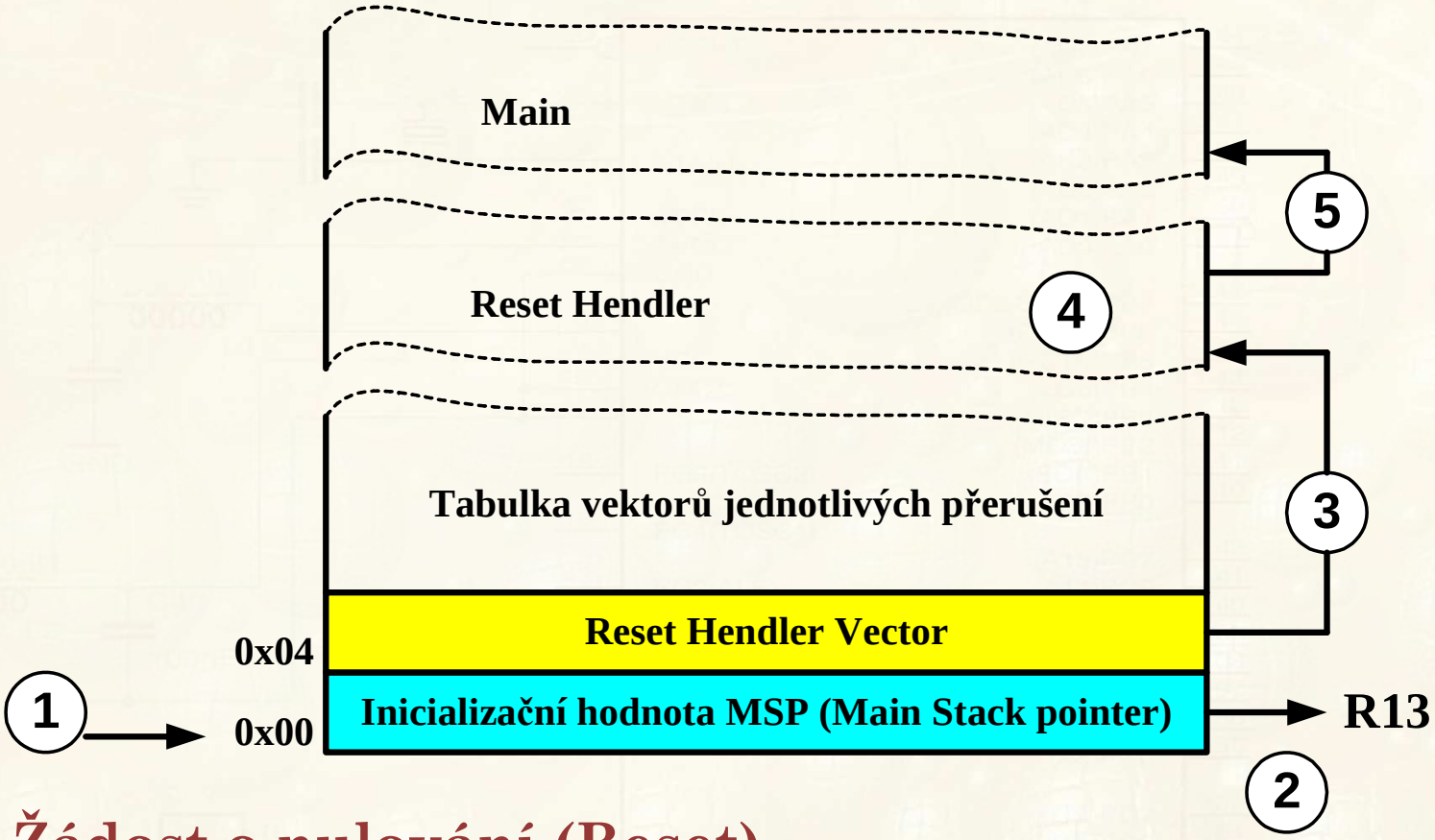
Prostor pro lokální proměnné

Knihovní funkce calloc(), malloc(), realloc(), free()

- **Stavový registr** - uchovává indikátory (příznaky) výsledku provedené operace. U ARM má označení **PSR**.

Standardní příznaky:	**u ARM**
Přenos	(označen **C**)
Částečný přenos	(není)
Přetečení	(označen **V**)
Parita	(není)
Znaménko	(není)
Záporný nebo menší než	(označen **N**)
Nula	(označen **Z**)
Saturace	(označen **Q**)

START PROCESORU ARM



1. Žádost o nulování (Reset)
2. Naplnění ukazatele MSP inicializační hodnotou
3. Přečtení adresy Reset Handler z adresy 0x04
4. Zpracování programu Reset Handler v Thread Módu
5. Přesměrování na Main program