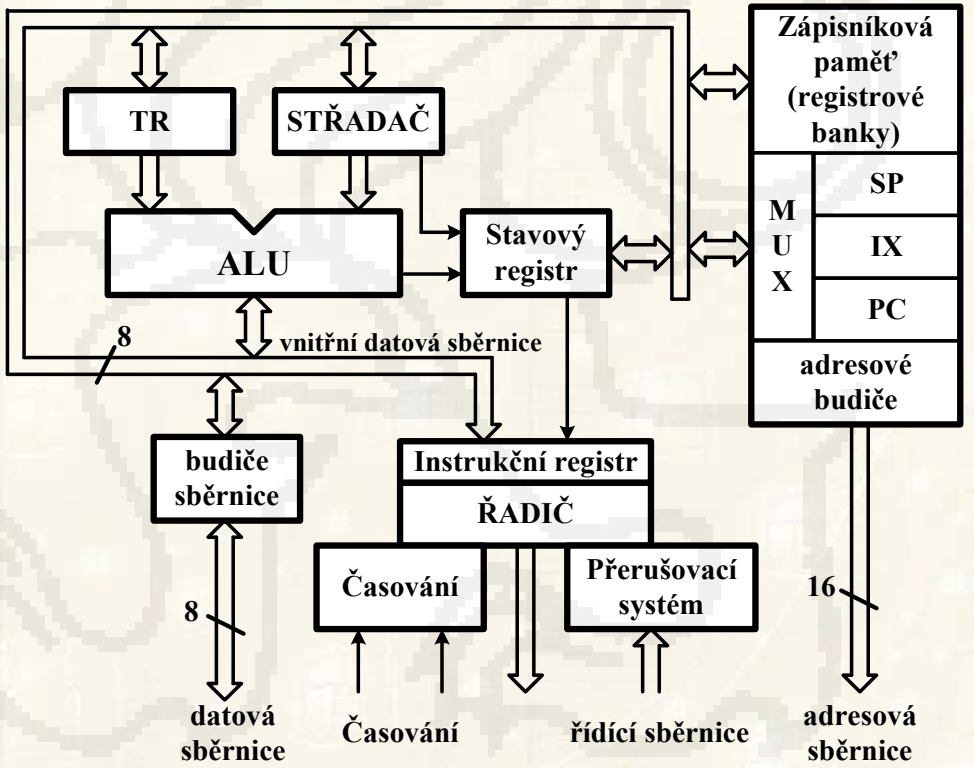
ARCHITEKTURA KLASICKÉHO MIKROPOČÍTAČE

První mikroprocesory byly vytvořeny z **registrů** (**A-střadače**, TR-dočasného registru pro uchování operandu při prováděné instrukci, **stavového registru pro uchování indikátorů** (flags, příznaků), IX-indexového registru pro indexové adresování, **PC-programového registru pro adresování následujících instrukcí v programu**, **SP-ukazatele zásobníkové paměti** situované do vnější datové operační paměti a registry **zápisníkové** paměti (u 8080 registry B, C, D, E, H, L), **aritmeticko-logické jednotky ALU, řadiče** a budičů datové, adresové a řídicí sběrnice. Vlastní procesor byl doplněn o generátor hodin a systémový řadič vytvářející řídicí signály MEMR, MEMW pro přístup do paměti IOR, IOW pro přístup do vstupně/výstupního prostoru a INTA pro externí řadič přerušení. Přenos hodnot byl direktivně řízený, synchronizovaný jedno nebo více fázovým hodinovým signálem. **Obvody musely vyhovět danému časování případně byl vygenerován signál WAIT**, který způsobil **vkládání čekací stavů**. Výběr periferních obvodů určuje adresa, přenášená po jednosměrné adresové sběrnici. Řídicí signály, které jsou odvozovány z instrukce a stavového registru, generuje řadič mikroprocesoru na základě vybrané fáze synchronizačního hodinového signálu. **Aritmeticko-logická jednotka ALU** realizuje aritmetické a logické operace vytváří a případně je ovlivněna příznakovými bity uloženými ve stavovém registru. ALU musí být schopna realizovat základní čtyři operace: **aritmetické sčítání, jednotkový doplněk (inverzi bitů), logický součin (identifikaci hodnoty bitu) a podmíněné větvení na základě hodnoty nebo bitu.**



ARCHITEKTURA KLASICKÉHO MIKROPOČÍTAČE

Zápisníková paměť je velmi rychlá paměť s malou kapacitou, která převážně slouží k **uchování operandů a adres** ukazujících na konkrétní místo v paměti. Operace s registry v zápisníkové paměti probíhají velmi rychle, protože se obvykle jedná o instrukci z jednoho bytu nebo slova, v kterém jsou registry implicitně identifikovány jen několika bity. Registry v zápisníkové paměti můžeme rozdělit na obecně použitelné a speciální, jejichž použití a využití je dáno výrobcem. **Střadač** je registr v němž je obvykle uložen jeden operand prováděné aritmetické nebo logické operace a do kterého se obvykle ukládá výsledek provedené operace. Má-li střadač a ALU větší počet bitů než případně vstupující operandy do ALU, potom musí být známo, zda zpracovávaná hodnota je bez znaménka nebo se znaménkem. **Řadič** řídí činnost spojenou s interpretací zpracovávané instrukce tj. vykonává posloupnost elementárních operací a generuje potřebné signály v závislosti na vykonávané instrukci včetně generování adresy a řídícího signálu pro načtení následující instrukce. Činnost řadiče můžeme rozdělit do těchto fází:

- Načtení instrukce (fetch) a její uložení do registru instrukcí
- Dekódování instrukce a inkrementace čítače instrukcí. Na základě dekódování je rozhodnuto o další fázi nebo opětovném čtení z programové paměti což může být součástí další fáze.
- Příprava operandů získává informace z dekodéru instrukcí, protože operandy prováděné operace mohou být uloženy v registrech, datové paměti nebo jako konstanty v paměti programové.
- Poslední fází je vlastní vykonání instrukce a uložení výsledku do registru, datové paměti nebo vstupně/výstupní brány.

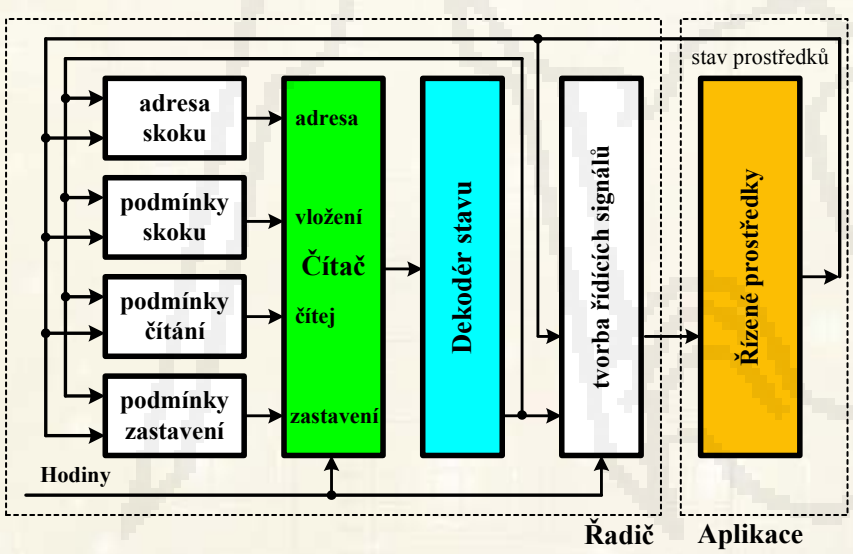
Řadič může být realizován:

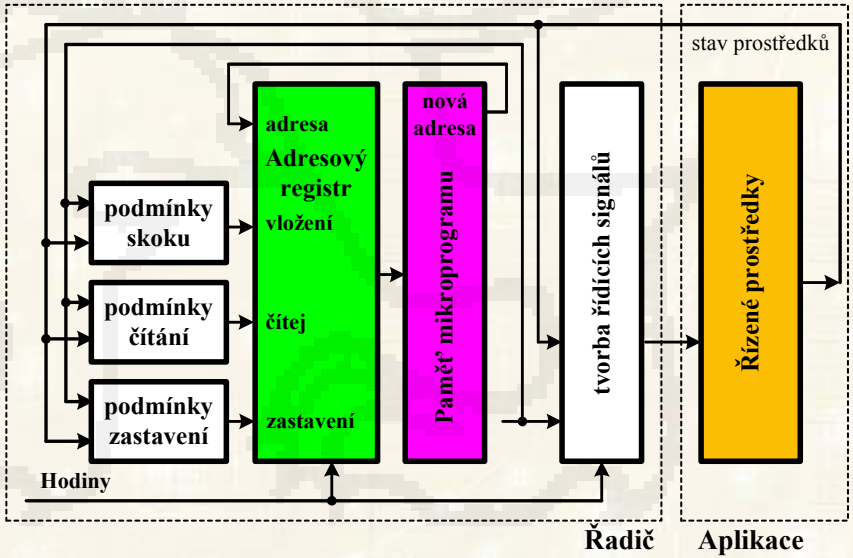
- Sekvenčním obvodem s neměnnou logikou. Tuto variantu využívají obvykle procesory RISC (Reduced Instruction Set) tzn. signálové a některé jednočipové procesory.
- Programový (mikroprogramovatelný) řadič u kterého se může nejen měnit logika, ale efektivně realizovat obsluhu velkého množství instrukcí za pomoci **mikrokódu**. Využívá se obvykle u procesorů CISC (Complex Instruction Set) s větším počtem instrukcí, které nemají stejnou dobu k vykonání.

ŘADIČ

Řadič je logický sekvenční obvod, který řídí činnost řízených prostředků i svoji vlastní (přechody mezi stavy). Automaticky přechází do následujících stavů, jestliže neobdrží od řízených prostředků žádost o čekání na vstupní podmínku nebo o provedení skoku. Základní struktury obvodového a programovatelného řadičem, jejichž činnost je synchronizována hodinovým kmitočtem, jsou zobrazeny na obrázcích. **Obvodový řadič** je tvořen čítačem, který inkrementuje svůj obsah (přechází ze stavu S_i do S_{i+1}), umožňuje zápis nového stavu S_k k realizaci skoku a může setrvávat v definovaném stavu do splnění určité podmínky. Na výstupech čítače je dekodér stavu, který generuje jednotlivé řídicí signály (často na kód 1 z N). Čítač může být nahrazen posuvným registrem s jednou rotující log.1. Činnost obvodového řadiče je pevná a jakákoliv změna vede na změnu obvodového zapojení.

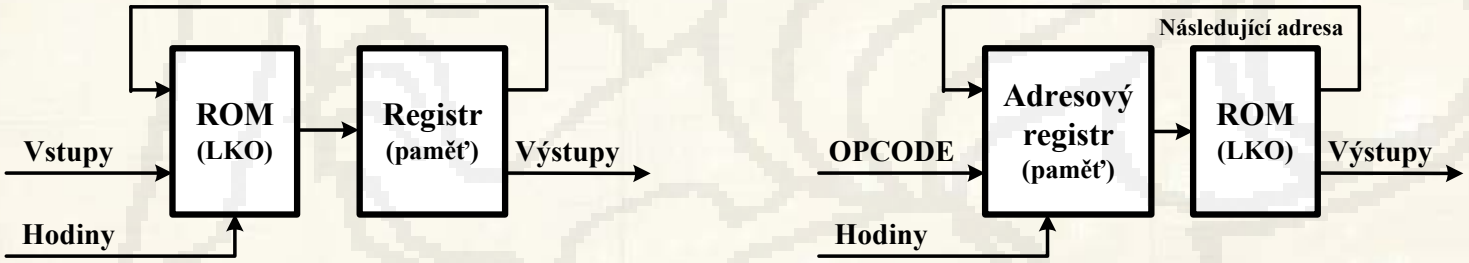
Mikroprogramovaný řadič v první fázi vznikl nahrazením dekodéru stavu pamětí PROM. Později byl upraven do konfigurace z obrázku, kde čítač byl nahrazen adresovým registrem (čítačem), který generuje adresy pro vybavení mikroinstrukcí uložených v paměti mikroprogramu. Mikroinstrukce obsahuje řídicí signály pro řízené prostředky, adresu následující mikroinstrukce a případně i další signály. Hlavní výhoda řešení spočívá v možnosti i velkých změn v činnosti řadiče bez zásahu do jeho obvodové struktury (bude-li vyhovovat počet vstupů a výstupů).





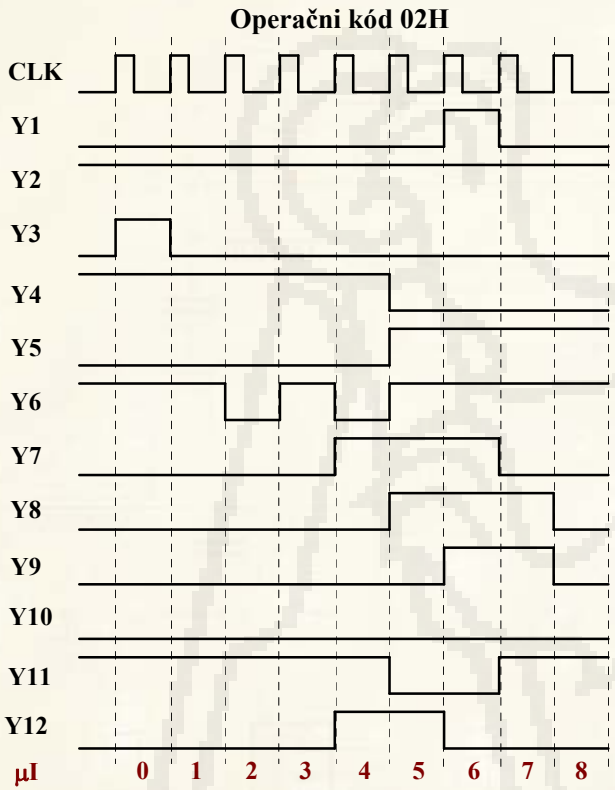
OD OBVODOVÉHO ŘADIČE K MIKROPROGRAMOVÉMU

Již v roce 1950 M.V.Wilkes ukázal, že mikroprogramování v podstatě reprezentuje logickou návrhovou techniku, která přináší řadu výhod oproti technice založené na Boolovských rovnicích a stavových diagramech. **Mikro-programování se od programování liší jen velmi málo** (mají obdobný formát instrukcí i programových struktur). **Rozdíl** lze vidět **hlavně v tom**, že **mikroprogramování podporuje návrh obvodového řešení** (hardware) a s **každou vykonanou instrukcí může generovat nový stav řídících signálů**. **Programování** (např. v jazyce symbolických adres) je spíše spojováno s manipulací s abstraktními daty a **řídící signály generuje obvykle po vykonání několika instrukcí**. Jednou z **realizací logického sekvenčního obvodu je tzv. automodifikace registru**, která představuje klasický návrh sekvenčního obvodu soustředěný do dvou obvodů. Registr realizuje paměťovou funkci obvodu a paměť ROM (PROM) realizuje kombinační část obvodu.



Obraťme nyní pořadí obou obvodů a předpokládejme, že výstupy LSO představují sériově vysílaná binární slova, která jsou čtena z paměti ROM a představují signály ovládající zařízení. Ve struktuře nedošlo jenom k záměně registru a paměti ROM, ale i **významové změně registru** (adresový) a vracejících se vodičů z paměti ROM budou představovat **následující adresu**. Následující adresa spolu s vstupem OPCODE, vytvářejí následující adresu z které se bude číst jeden stav generované sekvence (mikroinstrukce) odpovídající operačnímu kódu OPCODE (mikroprogramu). Současně s generovaným stavem bude generovaná nová adresa mikroprogramu odpovídajícímu přivedenému operačnímu kódu OPCODE. Na následujícím obrázku je zobrazena generovaná posloupnost a k ní odpovídající obsah uložený v paměti ROM s tím, že zpětné vodiče jsou přivedeny na adresové vodiče A3÷A0 paměti ROM a zbývající vodiče An÷A4 představují OPCODE. Výstupy Q3÷Q0 jsou zpětné vodiče a Q16÷Q4 představují výstupy Y12÷Y1.

OD OBVODOVÉHO ŘADIČE K MIKROPROGRAMOVÉMU



Adresa	Obsah ROM	Adresa	Obsah ROM
20h	42E1h	25h	8F26h
21h	42A2h	26h	1F37h
22h	40A3h	27h	5B28h
23h	42A4h	28h	4329h
24h	C4A5h		

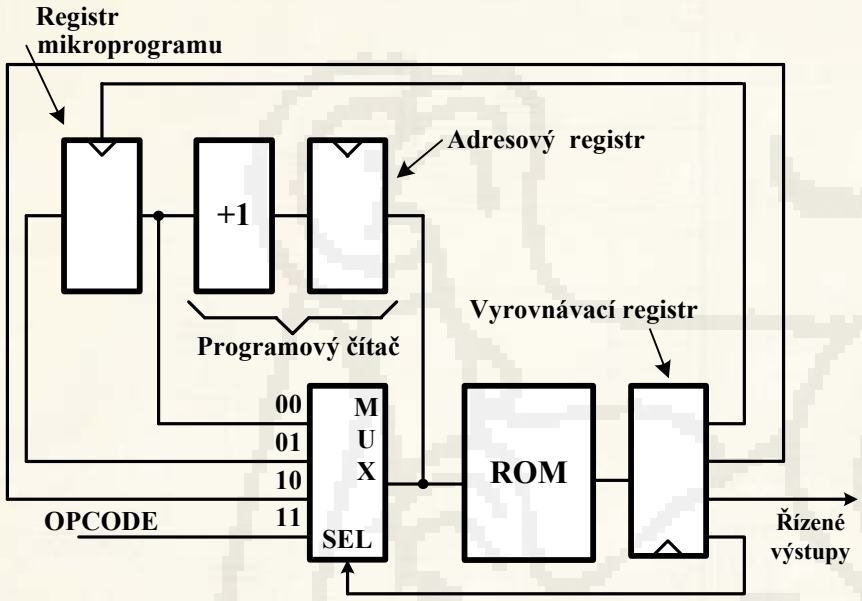
Obsah paměti ROM pak může mít následující význam

	Adresa ROM	Význam obsahu paměti ROM
Operační kód 00	00h až 0Fh	Mikroprogram pro instrukci DEC
Operační kód 01	10h až 1Fh	Mikroprogram pro instrukci ADD
Operační kód 02	20h až 2Fh	Mikroprogram pro instrukci SUB
	atd.	atd.
Operační kód 20	200h až 20Fh	Mikroprogram pro instrukci MOV

Z tabulky je zřejmé, že každému OPCODE přiřazen mikro-program **skládající se z 16-ti mikroinstrukcí** (případně by mohla obsluha některých OPCODE být i kratší). Tato sekvence není běžnému uživateli procesoru přístupná, ale je vyvolána pomocí operačního kódu (OPCODE) programu čteného z operační paměti procesoru (makroprogram).

Uvedená konfigurace umožňuje pro daný počet instrukcí vytvořit odpovídající mikroprogramy (např. o 16 mikroinstrukcích). Každý **mikroprogram bude muset obsahovat i sekvenci pro načtení další-ho operačního kódu (fetch)**, což u počítačů s velkým počtem instrukcí zbytečně zvětšuje velikost paměti ROM a degraduje řadič. **Efektivnější cestou by byla realizace skoku** do **mikropodprogramu (fetch),** z kterého se vrátíme do místa za mikroinstrukci, která jej vyvolala. Struktura z předcházejícího obrázku nám však operaci zavolání podprogramu neumožňuje. Modifikujme proto strukturu podle následujícího obrázku.

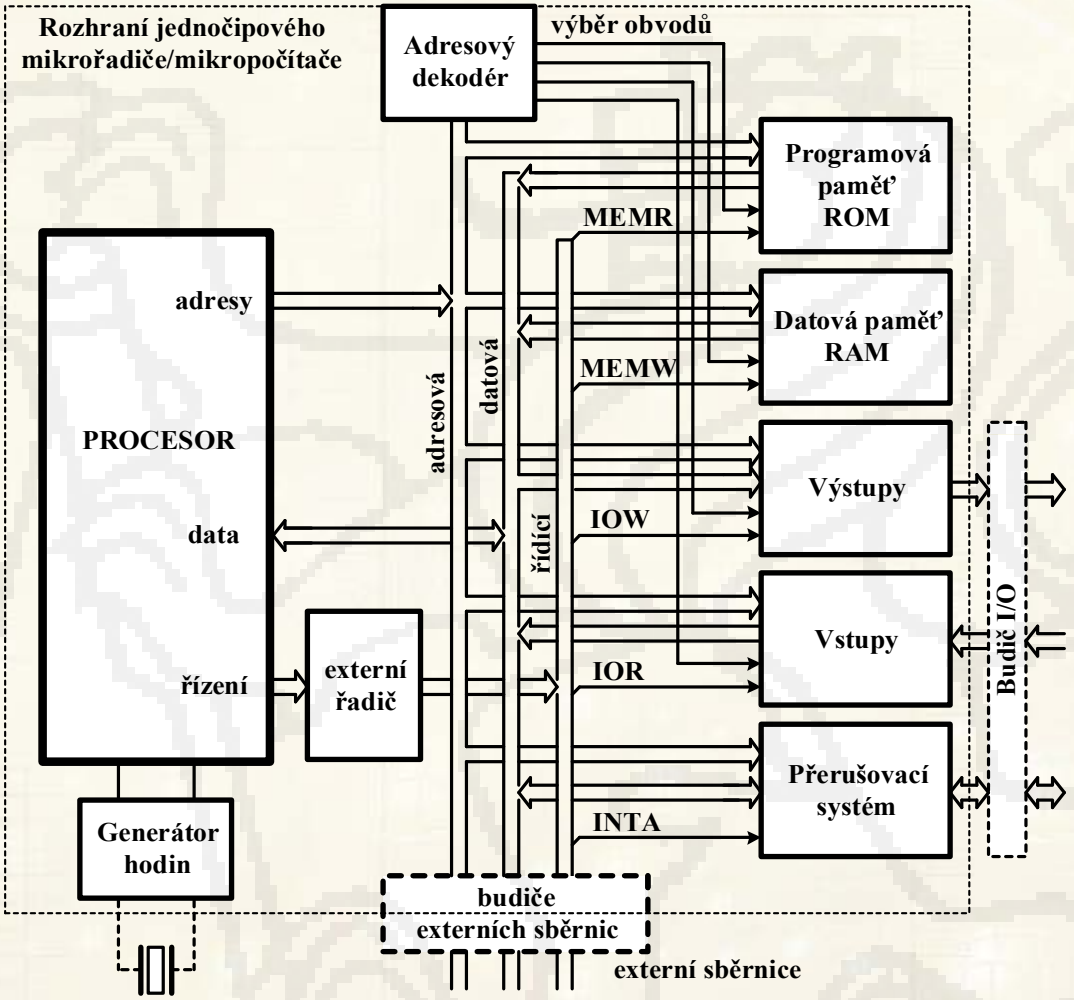
OD OBVODOVÉHO ŘADIČE K MIKROPROGRAMOVÉMU



Struktura z obrázku má tyto vlastnosti:

- **Registr mikroprogramu**
  Uchovává adresu následující mikroinstrukce, která bude vykonána po návratu z volaného mikropodprogramu.
- **Adresový multiplexer** umožňuje naplnění adresového registru obsahem registru mikroprogramu (návrat z podprogramu, RET, SELMUX=01), počáteční adresou mikroprogramu (zpracování nového OPCODE, JMP OP, SELMUX=11), adresou následující mikroinstrukce (sekvenční zpracování mikroprogramu, SELMUX=00) a adresou skoku (skok na mikroinstrukci nebo volání podprogramu, JMP ADR, SELMUX=10). Adresový registr byl přemístěn do větve vytvářející následující adresu a byl tak vytvořen adresový (programový) čítač. Přesunem bylo navíc odstraněno zpoždění operačního kódu o jednu periodu hodin. Vyrovnávací registr synchronizuje výstupy z ROM se synchronizačním hodinovým signálem. Navržená jednotka umožňuje sekvenční čtení mikroinstrukcí, volání podprogramu (jednoho) a návrat z nich a skok na libovolnou adresu v paměti ROM. Jednotka nemá implementovánu instrukci podmíněného větvení, tj. schopnost po sekvenci vykonaných mikroinstrukcí provést rozvětvení na základě stavu vnější nebo vnitřní podmínky. Změnou ovládání multiplexeru, lze vnější podmínku ignorovat nebo v závislosti na její hodnotě provést následující instrukci nebo skok na definovanou adresu. Jednotka neumožňuje realizaci programových smyček s proměnnou hodnotou jejich opakování. Pro funkci řadiče by bylo potřeba přidat stavové příznaky jako vstupní signály. Pro funkci mikroprogramovatelného řadiče pro rychlé řízení např. složité sběrnice byla tato struktura modifikována o podmíněné větvení a čítač, ale v dnešní době jako součástka může být snadno nahrazen programovatelným polem.

SKUPINOVÉ ZAPOJENÍ KLASICKÉHO MIKROPOČÍTAČE



MEMR - řídící signál (procesor čte data z paměti)
MEMWR - řídící signál (procesor zapisuje data do paměti)
IOR - řídící signál (procesor čte data ze vstupní brány)
IOW - řídící signál (procesor zapisuje data na výstupní bránu)
INTA - řídící signál (procesor čte vektorovou adresu z přerušovacího systému

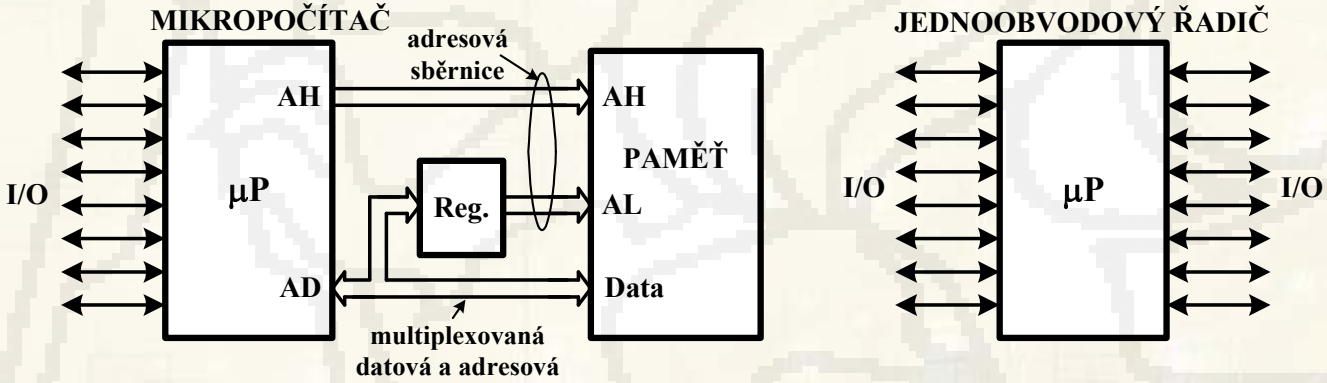
JEDNOOBVODOVÉ MIKROPROCESORY A ŘADIČE

Integrací procesoru s podpůrnými obvody a oscilátorem spolu s pamětí programu, dat, vstupů, výstupů a přerušo-vacího systému na jeden čip vznikl (1976) první jednočipový mikroprocesor z řady Intel MSC-48 (8048). Přestože 8048 byl v roce 1977 nahrazen dodnes populárním Intel 8051, jeho použití v PC klávesnicích, dálkových ovladačích a hračkách přispělo k jeho nízkým nákladům a široké dostupnosti. Vývoj jednočipových procesorů se do roku 2000 prakticky neměnil, standardní jádra procesorů inovovaná pouze technologicky byla doplňována určitým počtem V/V vývodů, interních periferií, typem a velikostí vnitřních pamětí. Výjimkou byl integrovaný koprocesor, změněné časování μP nebo 10-bitový A/D převodník. Proto řada dosud komerčně úspěšných procesorů má základ v procesorech z druhé poloviny sedmdesátých let minulého století. Jsou to například procesory odvozené z procesoru I8031/51 firmy Intel a procesoru MC6801 firmy Motorola (jedná se o řady MC6805, 68705 a 68HC11xx). Protože **„staré“ typy procesorů jsou neustále doplňovány novými obvodovými řešeními nestárnou**. **Koncepčně modernější jednočipové procesory** jako je SAB 80C166 firmy Infineon (Siemens), riskové procesory a řadiče ATmega a ATtiny od firmy Atmel, procesory PICxxxx od firmy Microchip **se na trhu prosazují proti tradičním starším procesorům pouze výkonností, cenou dobrým servisem výrobců**, kteří poskytují uživatelům zdarma vývojové prostředky a knihovny zdarma. S nástupem největších výrobců signálových procesorů (Analog Devices a Texas Instruments), kteří inovovaly procesory z řady 8051 a doplnily je 12, 16 a 24 bitovými A/D převodníky se vývoj v oblasti jednočipových procesorů výrazně změnil. Řada dalších firem (Motorola, Philips, Atmel, Cygnal, Dallas, Infineon, atd.) začala razantně inovovat svoje produkty. Rozvoj se u klasických mikroprocesorů soustředil na velikostí interních pamětí a změny v oblasti časování až k výkonu 100MIPS. Zároveň dochází k výraznému rozvoji RISC mikroprocesorů např. 32-bitových ARM7, ARM9, ARM11 - Advanced RISC Machine nebo aplikačně orientované procesory nebo multiprocesory např. OMAP35x v kterém je integrován jádro procesoru Cortex-A8 a signálové procesory TI C64x + DSP.

JEDNOOBVODOVÉ MIKROPROCESORY A ŘADIČE

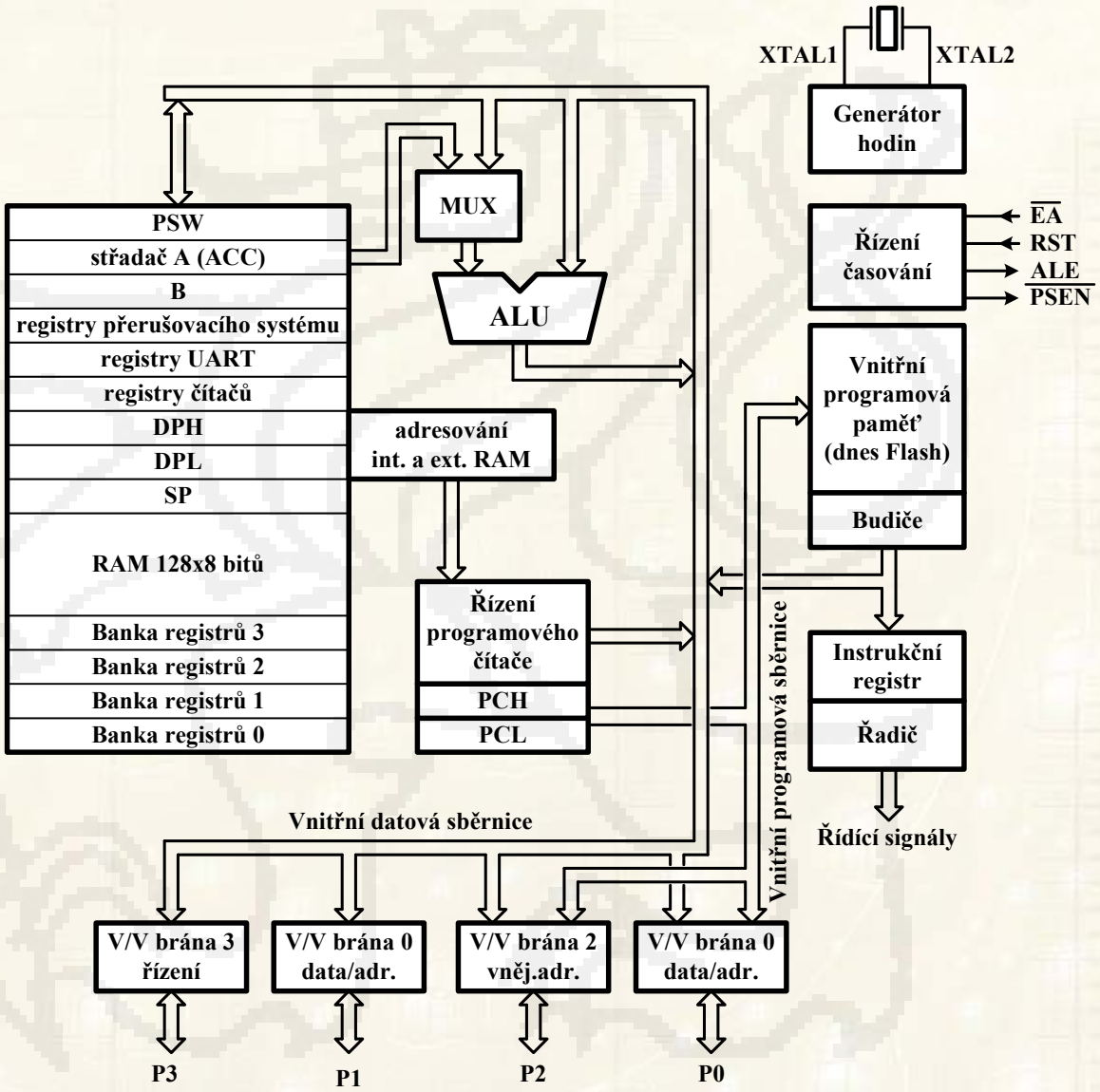
Historicky se ustálilo rozdělení jednočipových procesorů na **jednoobvodové mikroprocesory,** u kterých je možné vstupně/výstupní vývody využívat k připojení periferií nebo k **vytvoření společné sběrnice** pro připojení vnějších pamětí, V/V bran, převodníků, displejů, atd. **Periferie připojené k vnější společné sběrnici mohou být mapovány v programovém, datovém nebo vstupně/výstupním prostoru a ovládány příslušnou obvykle jedinou instrukcí.**

U **jednoobvodových řadičů nelze vytvořit společnou sběrnici ovládanou jedinou instrukcí.** Neznamená to, že řadič nemůžeme rozšířit o periferie nebo na jeho vývodech vytvořit společnou sběrnici, **které budou obsluhovány programem prostřednictvím několika instrukcí pro práci s I/O vývody mikrořadiče.**



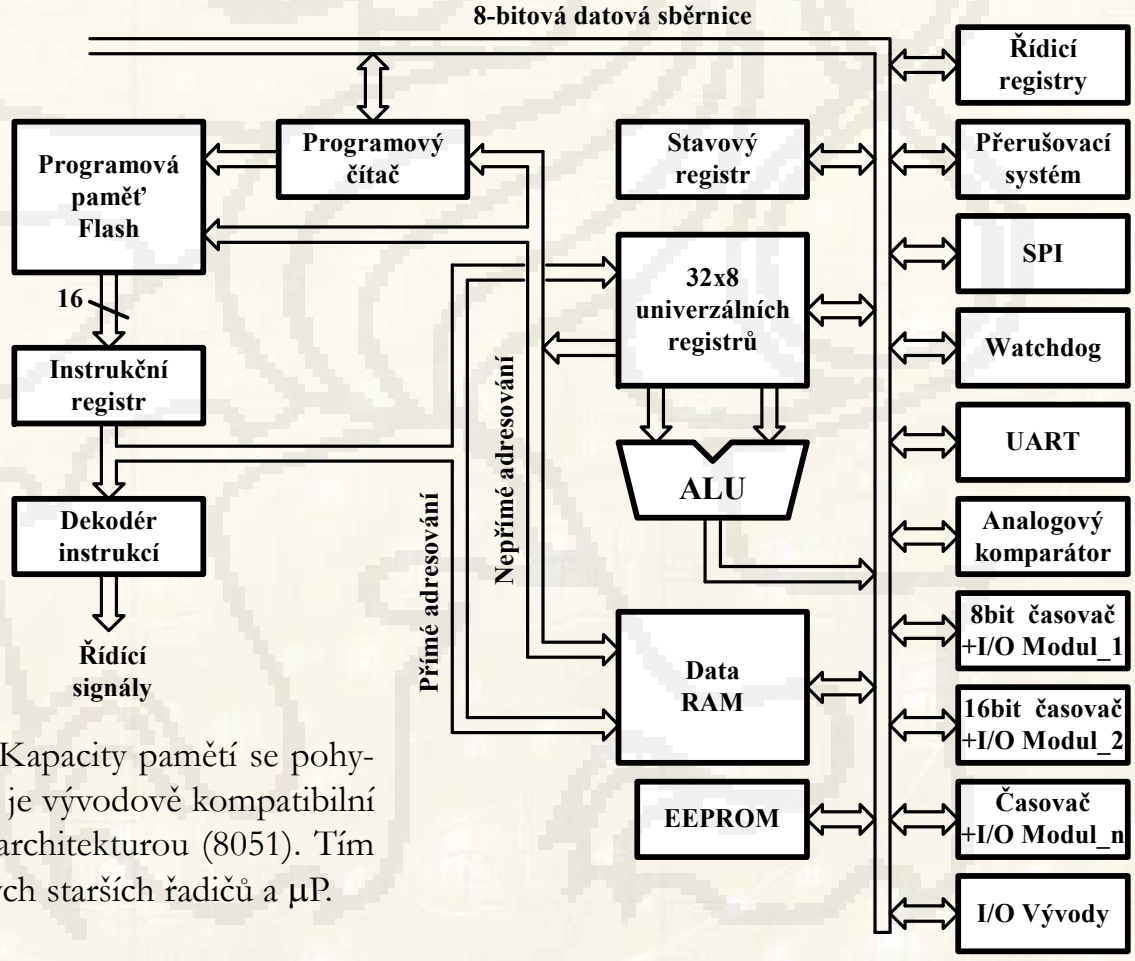
ARCHITEKTURA JÁDRA PROCESORŮ 8051

Mikroprocesor 8051 je osmibitový jednočipový mikroprocesor se smíšenou harwardskou a Von Neumanovskou architekturou, u které je oddělena programová a datová paměť, ale formát instrukcí a dat je stejný a přenáší se po stejné sběrnici. Procesor, jehož vnitřní struktura je blokově zobrazena na obrázku, je schopen samostatné činnosti po připojení vnějšího piezokrystalického rezonátoru („krystalu“) na vývody XTAL1 a XTAL2 a napájení 5V. Procesorová jednotka je tvořena ALU, která je doplněna násobičkou-děličkou a booleovskou aritmetickou jednotkou. **Pracovní registry R0 až R7, střadač A, registr B, stavový registr PSW a další jsou součástí interní přímo adresovatelné paměti RAM** (k **registru** se lze dostat přes **přímou adresu** nebo **implicitní instrukcí**). Nestačí-li interní programová nebo datová paměť, lze ji externě rozšířit až do přímo adresové kapacity 64kB.



ARCHITEKTURA JÁDRA PROCESORŮ ATmega

V polovině devadesátých let firma Atmel uvádí na trh řadu řadičů a mikroprocesorů s architekturou RISC s označením AVR (Enhanced RISC architecture). Procesory z této řady jsou vybaveny programovou pamětí Flash, kterou jde programovat přímo v mikropočítačovém systému s přenosem programovacích povelů a vlastního programu sériovým rozhraním SPI nebo dnes i JTAG. Procesory mají harwardskou architekturu s dvoufázovým překrýváním instrukcí (pipeline) a rozdílným formátem dat a instrukcí. Instrukce jsou 16-bitové a datová sběrnice je pouze 8-bitová. Procesory jsou vybaven násobičkou 8x8bitů se znaménkem, bez znaménka nebo desetinných. Procesory disponují okolo 130 instrukcí většinou vykonaných za jeden strojový cyklus. ALU spolupracuje se zápisníkovou pamětí obsahující 32x8bitů univerzálních registrů. Posledních šest registrů je konfigurováno jako tři 16-bitové registry s označením X, Y, Z. Každý zdroj přerušení má svoji adresu obslužného programu, což přispívá k rychlosti odezvy přerušení. Kapacity pamětí se pohybují od 4 do 128kB a řada z procesorů je vývodově kompatibilní s mikroprocesory a řadiči s intelskou architekturou (8051). Tím se firma snaží usnadnit přechod od svých starších řadičů a μP.



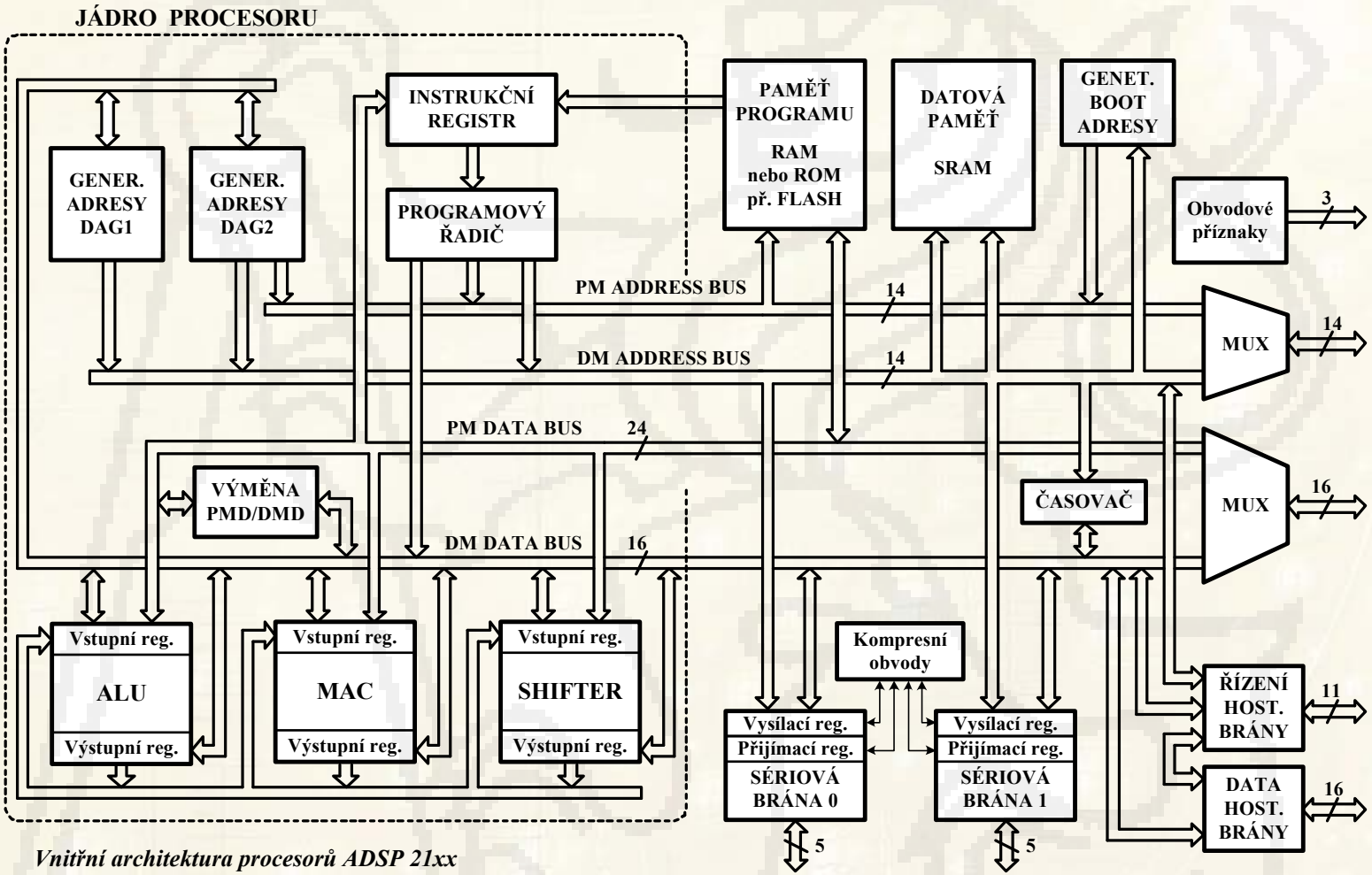
SIGNÁLOVÉ PROCESORY

První signálový procesor I2920, který byl původně určen pouze pro číslicové zpracování signálu, vyvinula v roce 1979 firma Intel. Procesor byl vybaven pamětmi s malou kapacitou, integrovaným A/D a D/A převodníkem. Hlavní nevýhodou byla nemožnost připojení vnější paměti a nepřítomnost násobičky. Procesor se neujal (Intel nepokračuje ve vývoji), ale ukázal dalším firmám možnou oblast vývoje. Nyní jsou signálové procesory využívány jak pro číslicové zpracování signálu a obrazu, tak i pro obecné logické řízení a regulaci. Důvodem je nízká cena úspěšných typů a mnohokrát větší výpočetní výkon, než mají i moderní a drahé univerzální procesory. **Architektura signálových proce-sorů je primárně podřízena co nejrychlejšímu zpracování základních operací** používaných v číslicovém zpracování signálu jako je

Konvoluce	$y_n = \sum_{i=0}^{M-1} h_i . x_{n-i}$	Filtrace	$y_n = \sum_{i=0}^{M} a_i . x_{n-i} - \sum_{i=1}^{L} b_i . y_{n-i}$
Diskrétní transformace	$X_k = \sum_{i=0}^{M-1} x_i . W^{ik}$	Korelace	$y_n = \frac{1}{N} \sum_{i=0}^{N-1} x_i . x_{n+i}$

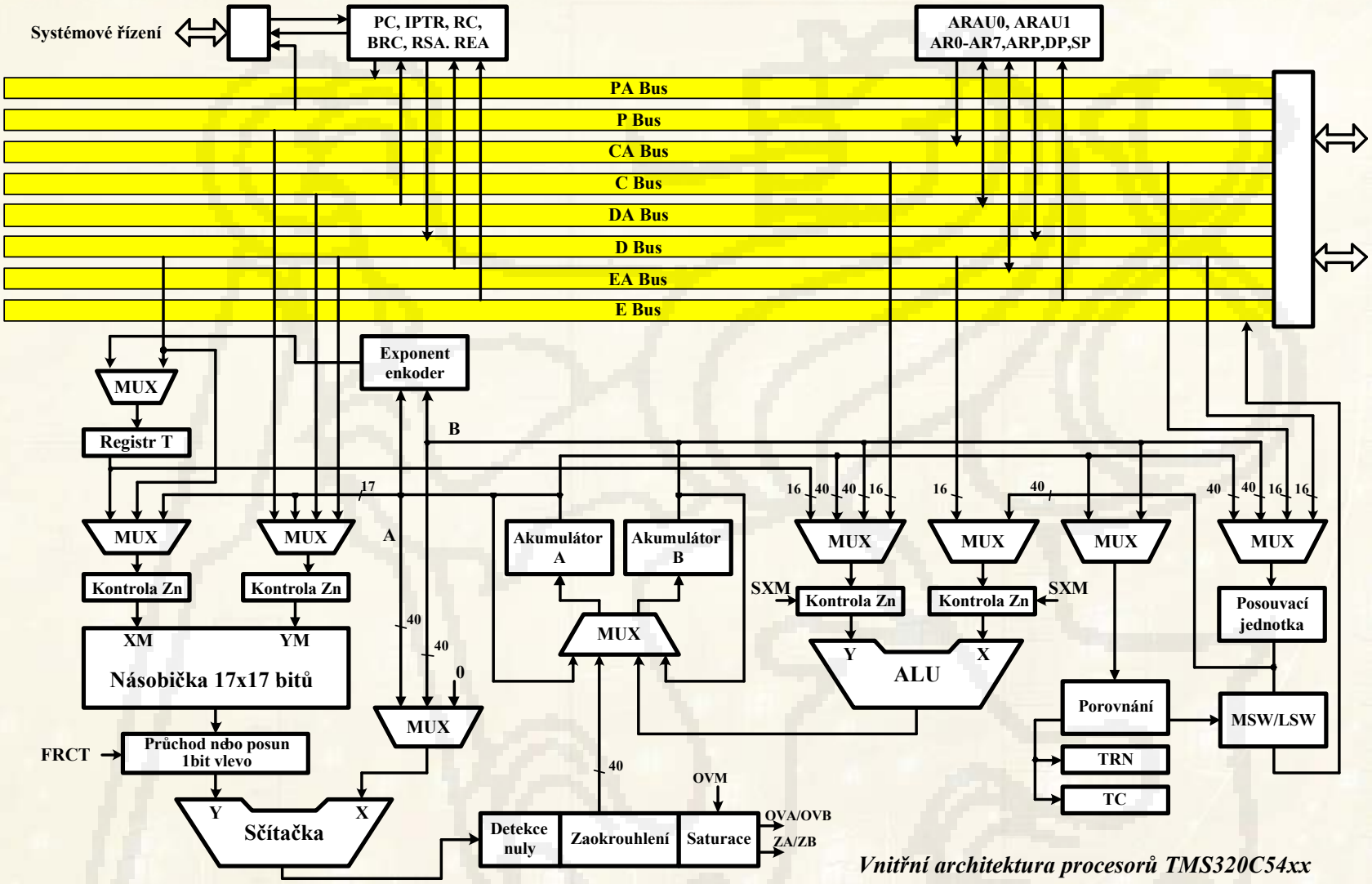
Ze vztahů je vidět, že **hlavní důraz musí být kladen** na operaci **násobení, sčítání (akumulace) a posunu**. Odtud architektura i instrukční soubor signálových procesorů je podřízen jejich co nejrychlejší realizaci. **Až do roku 1992** patřily **signálové procesory mezi nejrychlejší procesory** na světě a postupně začaly vytlačovat univerzální procesory z oblastí, kde je třeba vysoký výpočetní výkon (regulace, řadiče disků a tiskáren, atd.). V současné době se svým výkonem přibližují výkonu univerzálních procesorů pro PC, ačkoliv pracují s až 10x nižším hodinovým kmitočtem.
Obecně mají signálové procesory modifikovanou harvardskou struktury s odděleným programovým a datovým prostorem, který umožňuje **současnou manipulaci se dvěma nebo více operandy**. Struktura je modifikovaná proto, aby bylo možné realizovat **adaptivní algoritmy**, při kterých je potřeba realizovat přenosy nejen z programové paměti do datové, ale **i obráceně**. Procesory disponují redukovanou instrukční množinou s instrukcemi přizpůsobenými rychlé realizaci násobení a akumulace. Disponují větším počtem paralelně pracujících jednotek MAC (násobička +aritmeticko-logická jednotka), posuvná jednotka, generátory adres, aritmetická jednotka v oblasti PC s podporou obvodového čítání. Výpočetní výkon se nejprve zvyšoval zřetězením instrukcí princip „sdílení času“ (dnes 2 až 7 (skrytě 12)).

ARCHITEKTURA BĚŽNÉHO SIGNÁLOVÉHO PROCESORU



Vnitřní architektura procesorů ADSP 21xx

ARCHITEKTURA BĚŽNÉHO SIGNÁLOVÉHO PROCESORU

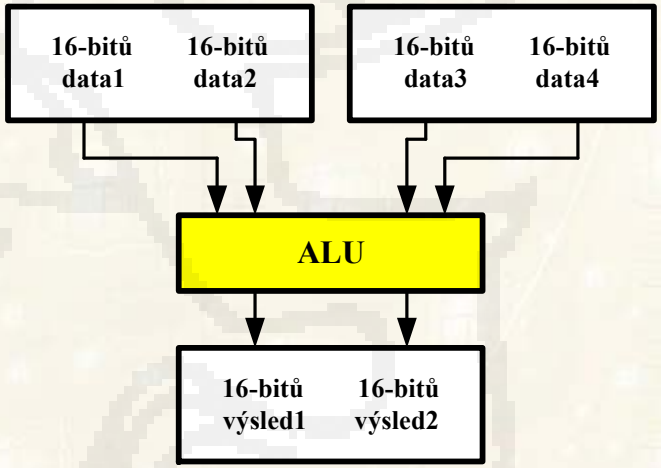


Vnitřní architektura procesorů TMS320C54xx

ZVYŠOVÁNÍ VÝPOČETNÍHO VÝKONU PROCESORŮ

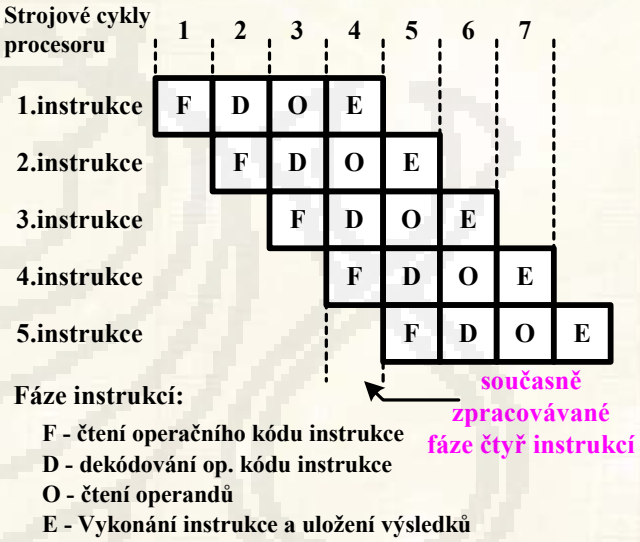
Zvyšování výpočetního výkonu u univerzálních, signálových a částečně i jednočipových procesorů probíhá v oblasti technologie (maximalizace hodinového kmitočtu), ale nyní hlavně v oblasti architektury procesoru. Vývoj překonal odhady maximálních kmitočtů pro jednotlivé technologie (bipolární Si - 800MHz, CMOS – 3,3GHz), ale v poslední době dochází k výraznému zpomalování zvětšování vzorkovacího kmitočtu na dvojnásobek za každých 18 měsíců. Architektura procesorů se soustředí na paralelizaci procesů a zpracování a díky tomu se výkonnost procesorů stále zvětšuje. Paralelizace operací se ubírá zvětšováním počtu paralelně pracujících jednotek **superskalární** architektury, u které paralelně prováděné instrukce zajišťuje obvodově řadič procesoru (např. PC). Druhou cestou paralelizace instrukcí, která se především uplatňuje u signálových procesorů, je architektura **VLIV**, kde současně zpracovávané instrukce zajišťuje překladač a ukládá je do dlouhého instrukčního slova již ve fázi překladu programu. Další variantou jsou paralelně řazené segmentované aritmetické jednotky tzv. superscalar super-pipelining (Alpha, P4). Zatím poslední variantou je proces paralelizace založený na zvětšování počtu jader (multicore). Uplatňované postupy **paralelizace**, které **mohou být uplatňovány na zpracovávaná data nebo instrukce** si popíšeme samostatně.

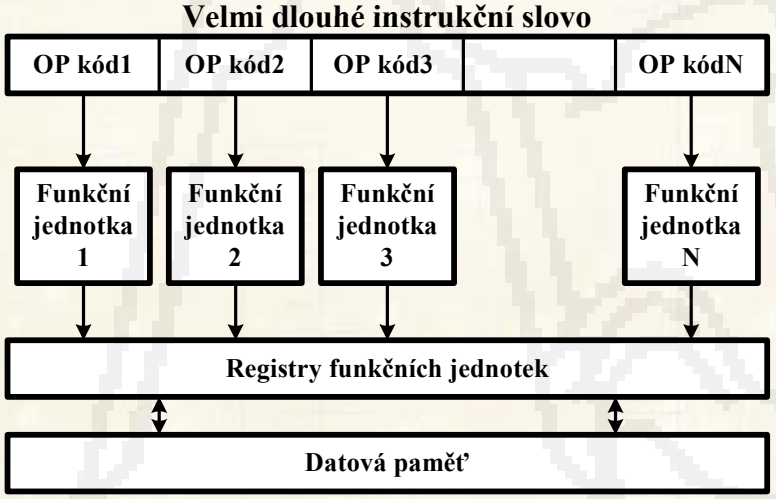
SIMD (Single Instruction stream, Multiple Data stream) je podpůrný prostředek využívaný od 90 let. m.s. ke zvýšení výpočetního výkonu procesorů MMX (PC) a u signálových procesorů. SIMD je systém zpracovávající několik datových toků jedním programem. Jedná se o koncepci, při které je jednou instrukcí realizována stejná operace na skupině nezávislých dat sloučených do jednoho delšího datového slova. Dnes jej využívají všechny typy současných signálových i PC procesorů.



ZVYŠOVÁNÍ VÝPOČETNÍHO VÝKONU PROCESORŮ

Princip **segmentace instrukce** do jednotlivých překrývajících se fází byl přirozeným vyústěním snahy o zvyšování výpočetního výkonu pomocí částečného překrývání instrukcí zahájeného již v 70.letech m.s. na procesoru 8051. Segmentovaná aritmetická jednotka nebo celý procesor má instrukci rozdělenu do dvou a nebo více částí. Výsledky dané fáze jsou uloženy do vyrovnávací paměti, z které si je v dalším strojovém cyklu přebírá ke zpracování další segment. Současně s tím může být zahájeno zpracování další instrukce. Konfigurace se označuje „Princip sdílení času" nebo pipeline. Je-li fází více jak 4 hovoří se „superpipeline", který výrazně komplikuje obsluhu operační paměti a větvení programu. Využívá se u univerzálních a signálových procesorů, u jednočipových procesorů se nyní používá maximálně 2 fázový pipeline.

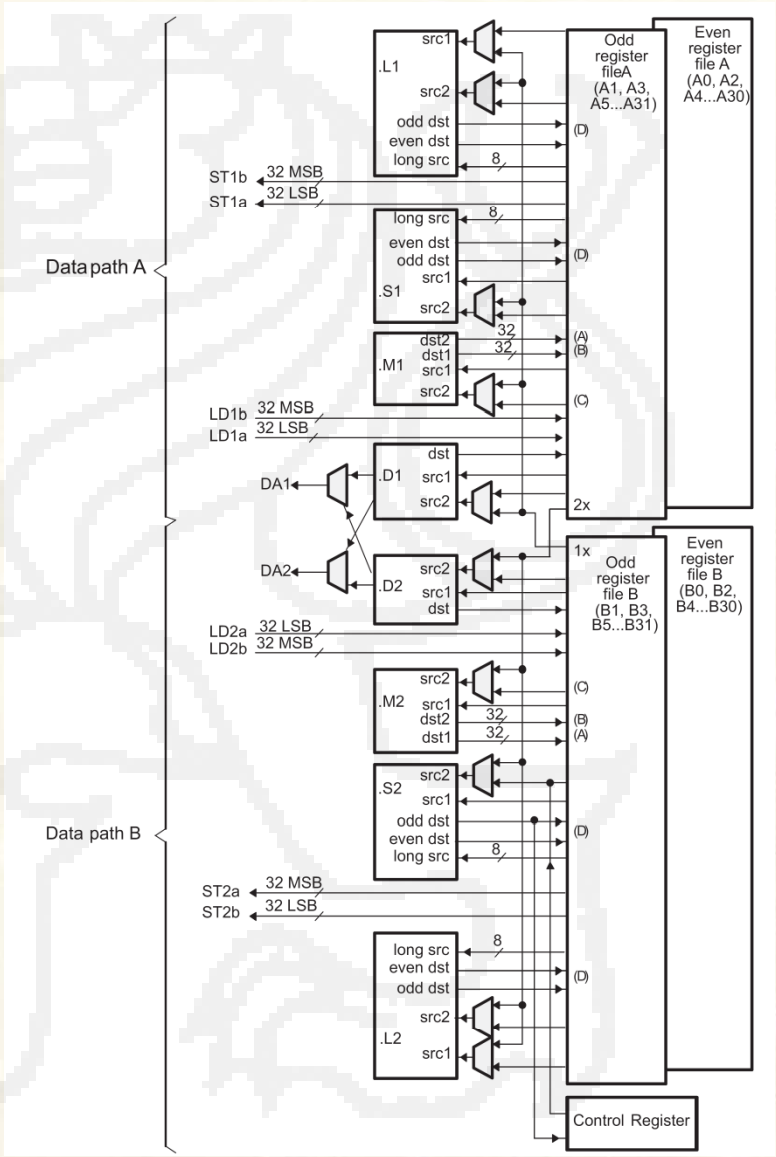




MIMD (Multiple Instruction stream, Multiple Data stream) je obecný typ paralelního systému zpracovávajícího několik datových toků samostatnými jednotkami, z nichž každá realizuje samostatný program. Pod princip MIMD můžeme zařadit multiprocesorové systémy (samostatné, integrované do čipu (pole procesorů TMS320-C8x, AD14060, atd.)). **Nyní se ve větší míře rozvíjí cesta zvyšování počtu aritmetických jednotek v jednom pouzdře** (WLIV a její modifikace - čtení několika instrukcí najednou pro každou jednotku).

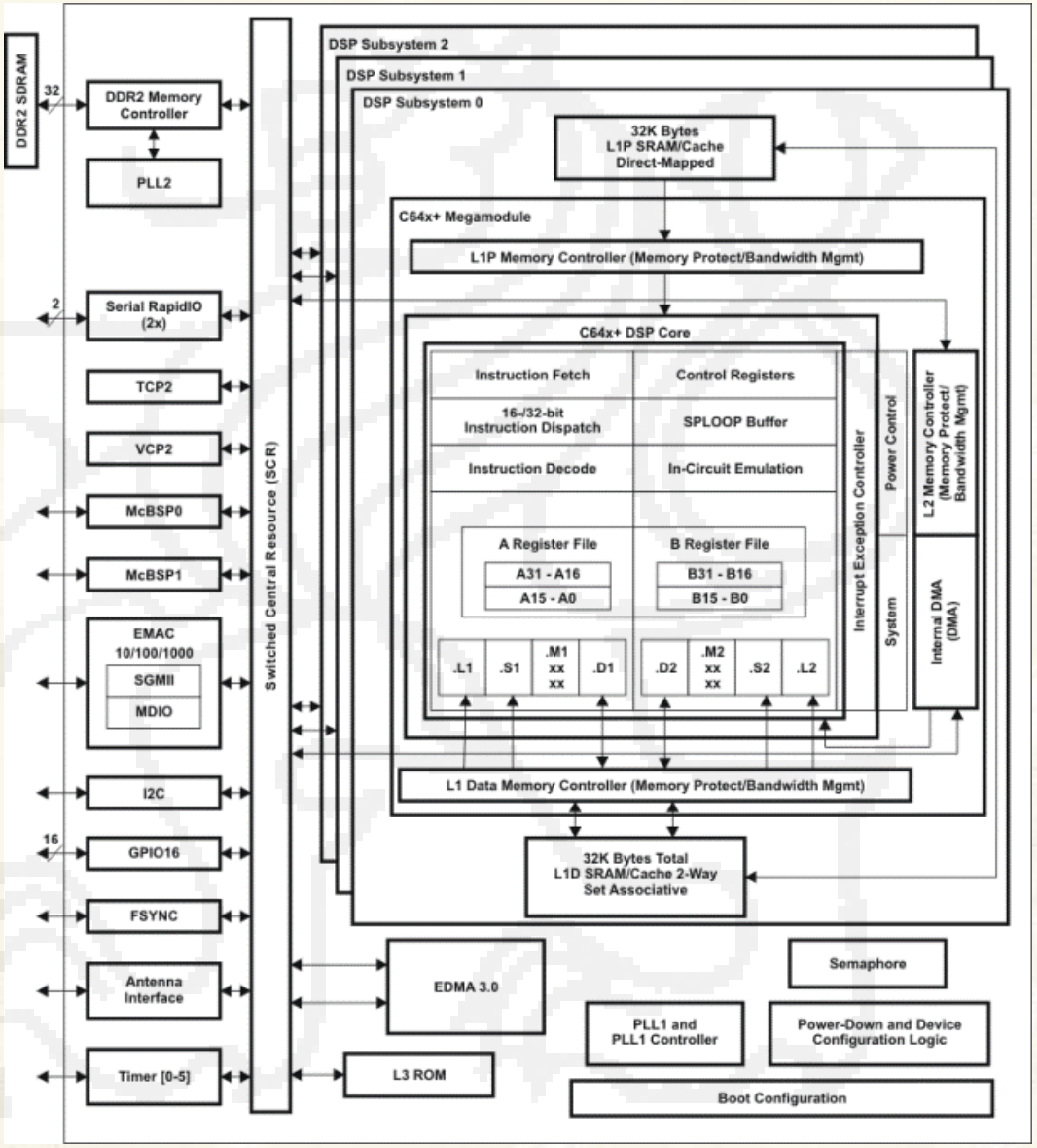
SIGNÁLOVÉ PROCESORY VLIW

Paralelizace u běžných signálových procesorů spočívá v tom, že sdružená instrukce a následně řadič paralelní chod MAC, ALU, Shifter, generátorů adres a aritmetické jednotky v oblasti čítače instrukcí. **Signálové procesory**, které čtou **více instrukcí** najednou, vyvinula a jako první firma Texas Instruments (1997) pod označením VLIW (Very Large Instruction Word). U procesorů tohoto typu se paralelně načte 256 nebo 128 bitové instrukční slovo, které se skládá z několika instrukcí pro jednotlivé paralelně pracující aritmetické jednotky. U procesoru TMS320C6xxx se jedná o 8 32-bitových instrukcí. Podobné je to i u konkurenčních procesorů (Agere System a Motorola nebo Analog Devices). Počet funkčních jednotek se mění výrobce od výrobce s tím, že následovníci TI se snaží inovovat nevýhodné vlastnosti původní struktury VLIW. V TMS320C64x jsou integrovány dvě násobičky a šest dost podobných aritmetických jednotek, z nichž dvě jsou předurčeny k adresování a zbývající jsou k volně dispozici. **Instrukce** pro jednotlivé funkční jednotky **mají** v sobě uchovánu informaci o **paralelním** nebo **sériovém vykonání** s dalšími instrukcemi. Snahou je, aby všechny jednotky vykonávaly v každém strojovém cyklu instrukci a možný výkon procesoru tak byl využit. Odtud problematika paralelního programování se stává dominantní pro tento typ procesorů. U procesoru TMS320C6x komplikuje skutečnost fakt, že instrukce se sice vykonají za jeden strojový cyklus, ale jejich výsledek je k dispozici až po 1, 4 nebo 5 strojových cyklech.



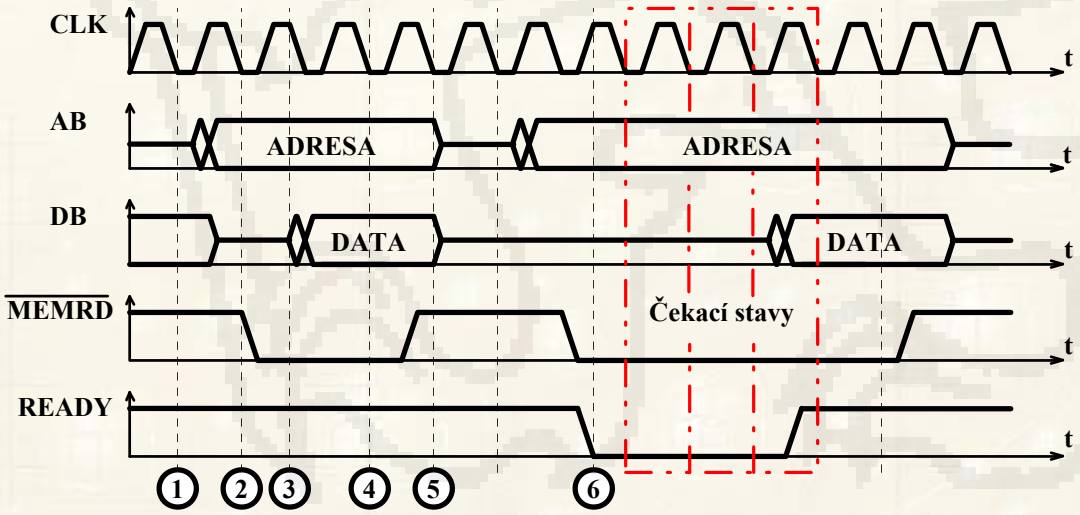
VLIW SIGNÁLOVÉ MIKROPROCESORY S NĚKOLIKA JÁDRY

V současné době nejvýkonějším signálovým procesorem je procesor TMS320 C6474, který v sobě obsahuje tři jádra procesoru VLIW z řady 64xx a disponuje při hodinovém kmitočtu 1,2GHz výkonem až 28800 MIPS. Procesor je vybaven rozhraním I2C pro kontrolu periferií nebo komunikaci s hostujícím procesorem, dvěma buffrovanými sériovými kanály McBSP0 a 1 s přenosem až 100Mb/s, 6-ti 64 bitovými nebo 12-ti 32 bitovými čítači, 16-ti V/V branami GPIO s programovatelným přerušením, Ethernet kontrolérem EMAC, 16/32 bitovým rozhraním pro DDR2, koprocesorem Vitherbiho dekodéru VCP2 a turbo kodéru TCP2, obvodovým semaforem, rychlými sériovými branami Serial RapidIO pro blízké propojení procesorů. Oproti jádru 64x byly jednotky M1 a M2 modifikovány tak, že jednom strojovém cyklu vypočtou jeden součin 32x32bitů, dva součiny 16x32 nebo 16x16 bitů, čtyři součiny 8x8 nebo 16x16 se sčítáním/odčítáním včetně jednoho komplexního násobení CMPY.



PŘENOS PO SPOLEČNÉ SBĚRNICI MIKROPOČÍTAČE

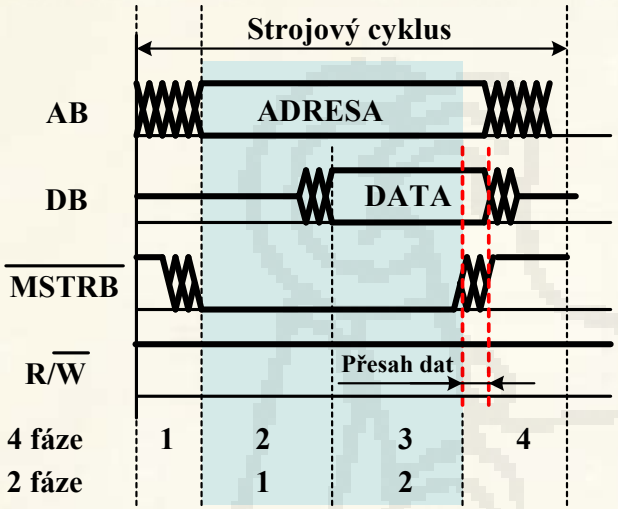
Na rozdíl od jednoduchých logických sítí, kde jeden vodič slouží k přenosu signálu od jednoho zdroje (výstupu log. členu) k jednomu nebo více vstupům, se v technice počítačů používá systém sběrnic, který vytváří efektivní spojení všech součástí uP systému. Část vodičů společné sběrnice je využívána pro přenos dat mezi bloky počítače – datová sběrnice, část pro přenos adres určujících, která část počítače je vůči procesoru zdrojem nebo příjemcem dat – adresová sběrnice a zbývající vodiče slouží k přenosu řídících signálů zajišťujících přenos dat po sběrnici – řídicí sběrnice. Vyjma přímého přístup do paměti DMA (Direct Memory Access) se přenos dat v mikropočítači odehrává vždy **mezi procesorem a určitým** (adresou a řídícím signálem definovaným) **blokem** (pamětí, vstupně/výstupním obvodem, atd.) a proto **datová sběrnice musí být obousměrná**. Naproti tomu **adresová i řídicí sběrnice jsou jednosměrné**. Adresa paměti nebo periferie je přivedena na adresovou sběrnicí ke zbývajícím částem procesorového systému. Řídicí signály určují zda prováděná operace bude vůči procesoru vstupní a výstupní. **K datové sběrnici mohou být připojeny** pouze výstupy **obvodů s otevřeným kolektorem** (dříve) nebo obvodů **s třístavovým výstupem** (nyní skoro výhradně). **V daném časovém okamžiku je datovou sběrnicí přenášena informace jedním směrem** a proto smí být aktivní pouze jeden třístavový výstup paměti nebo periferie (při přenosu do CPU) nebo žádný (při přenosu z CPU).



PŘENOS PO SPOLEČNÉ SBĚRNICI MIKROPOČÍTAČE

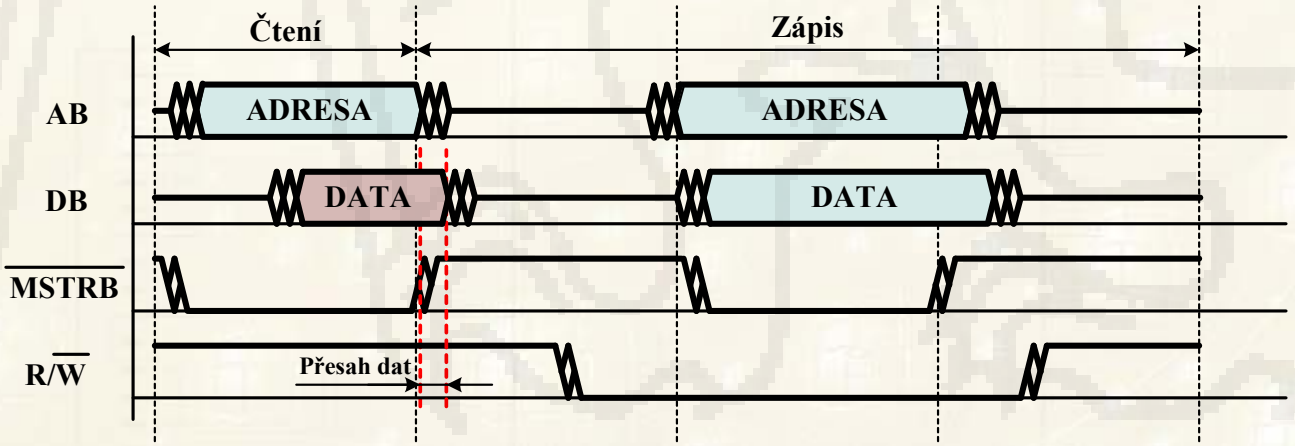
Řízení přenosu dat v mikropočítači je direktivní (bez potvrzení) a proto musí být každý obvod schopen v době určené časováním procesoru poskytnout nebo přijmout data. Pokud tento předpoklad splněn nebude jsou některé procesory vybaveny signálem WAIT, READY nebo generátorem čekacích stavů umožňující spolupráci s pomalou pamětí nebo periferií. Na předcházejícím obrázku je zobrazeno typické (čtyř fázové) taktování na společné sběrnici při čtení paměti bez čekání a s čekáním u blíže neidentifikovaného procesorového systému. **Časový průběh signálů** na sběrnici **při čtení jednoho slova** z paměti začíná v časovém okamžiku ①, kdy procesor vysílá na adresovou sběrnici adresu paměťového místa (buňky) na níž je přenášené slovo uloženo. S jistou časovou prodlevou se aktivuje řídící signál ②, který uvádí paměť do režimu čtení. Paměť dekóduje adresu spolu se svým aktivačním signálem a za **dobu vybavení dat** přivede na datovou sběrnici čtenou hodnotu ③, kterou procesor čte po ustálení poměrů na sběrnici (pro obvody s rychlým přechodem se bude vodič sběrnice chovat jako nesymetrické mikropáskové vedení s relativně malou charakteristickou impedancí 30÷100Ω) v časovém okamžiku ④. Po přečtení hodnoty ukončí procesor signál čtení, který způsobí deaktivaci paměti a přechod datové sběrnice do stavu vysoké impedance ⑤. Díky parazitním kapacitám přechází datová sběrnice postupně do úrovně odpovídající napětí na vstupech logických členů, které jsou na ni připojeny, nebo do úrovně odpovídající upnutí nebo přizpůsobení sběrnice. Existují sběrnicové obvody, které si na vstupech podrží poslední hodnotu, která byla na sběrnici, aby nedocházelo k oscilacím na sběrnici. Po ukončení signálu čtení ukončuje procesor platnost adresy přechodem do stavu vysoké impedance nebo přivedením nové platné adresy. V druhé části obrázku je zobrazena situace v případě přístupu k pomalejší paměti, kdy v okamžiku platné adresy a řídícího signálu je od periferie do procesoru vyslán signál READY (WAIT) ⑥, který způsobí vkládání čekacích stavů. Vlastní signál musí být aktivní tak dlouho, aby paměť byla schopna vybavit nebo přijmout informaci. Po uvolnění signálu probíhá zbývající část rozdělané instrukce. Obdobně vypadají časové průběhy při zápisu s tím rozdílem, že po platné adrese vysílá procesor nejprve platná data a teprve po jejich ustálení na datové sběrnici generuje řídící signál pro zápis do paměti nebo V/V zařízení. S nástupem **výkonných procesorů** s krátkým strojovým cyklem, začaly někteří výrobci přistupovat k **dvoufázovému přístupu** do pamětí a V/V jednotek, aby **zvětšili časový prostor pro dobu vybavení paměti** (pomalejší paměti budou levnější).

PŘENOS PO SBĚRNICI S 2 A 4 FÁZOVÝM PŘÍSTUPEM



V tabulce je zobrazen poměr doby vybavení pro běžné signálové procesory od firmy Texas Instruments od první až po pátou generaci. Přechod na dvoufázový paměťový interface dovoluje použití pamětí s delší dobou přístupu a snižují se tak náklady na materiál. Zavedení dvoufázového přístupu způsobuje zavedení tří cyklového zápisu mezi dvěma operacemi čtení. Ztráta zápisem u filtru FIR 20 stupně je cca 4% u filtru 100 stupně 1%. Pro typické operace číslicového zpracování (akumulace součinů) se přechod na dvoufázový strojový cyklus příliš neprojeví.

Procesor	Fáze	S.cyklus	SRAM	Poměr
C10	4	200 ns	75 ns	37,5%
C25	4	100 ns	35 ns	35%
C50	2	50 ns	32 ns	64%
C54x	2	25 ns	15 ns	60%



PŘERUŠOVACÍ SYSTÉM

Pro obsluhu vnitřních i vnějších periferií je mikroprocesor vybaven přerušovacím systémem (**Interrupt system**), který podstatně zjednodušuje programové řízení styku s periferiemi. Jedná se o takový „zvonek“, který procesoru oznámí, že je potřeba obsloužit určitou periferii. Pokud by procesor tímto systémem vybaven nebyl, musela by v hlavním programu část kódu být věnována testování připravenosti těchto periferií. **Testování by nejenom snižovalo „výpočetní výkon procesoru“ pro realizaci hlavního programu, ale navíc by nemuselo zaručovat zpracování v potřebných časových intervalech.** Periferie využívající přerušovací systém svou žádostí zajistí ve vhodném okamžiku přerušení probíhajícího programu a procesor přejde ke zpracování **obslužného programu**. Obslužný program je vyvolán obvodově obvykle pomocí instrukce **LCALL adresa_přerušení**. Vyvolání přerušení je doprovázeno posloupností operací:

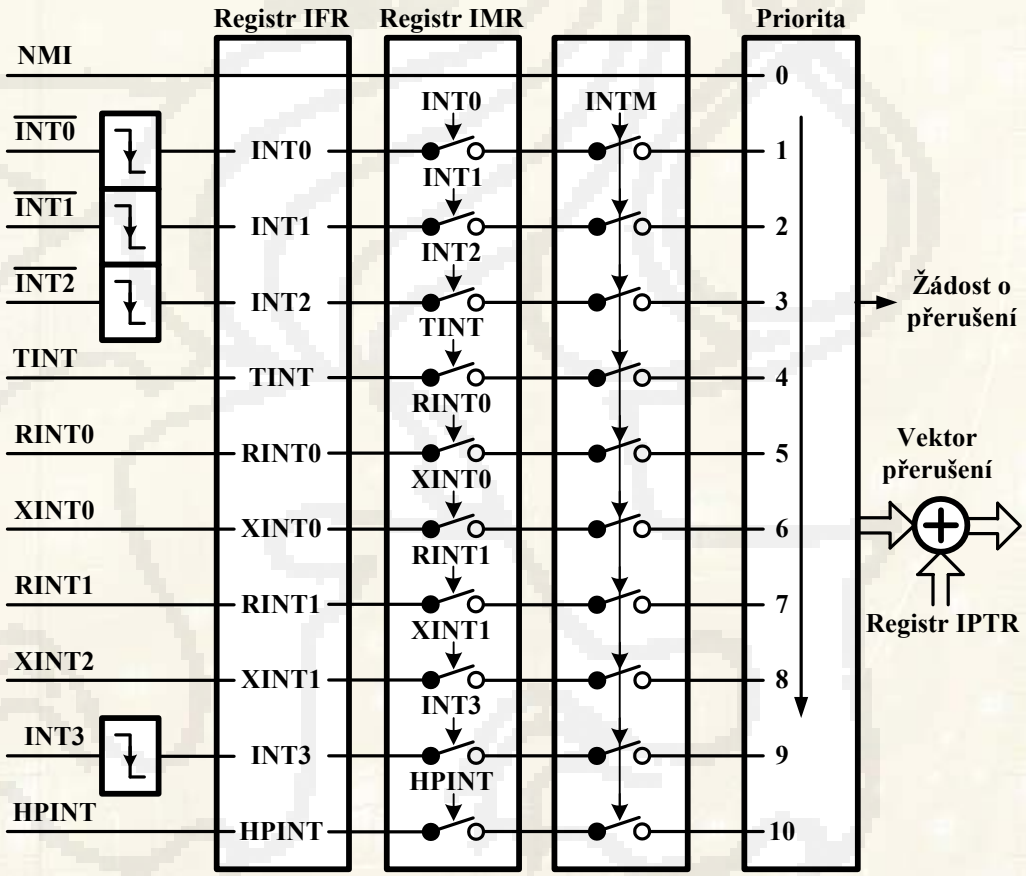
- Procesor dokončí právě rozpracovanou instrukci nebo instrukce („pipeline“)
- Uloží obsah čítače instrukcí do zásobníkové paměti (adresu následující instrukce po poslední vykonané)
- Nastaví čítač instrukcí na pevně určenou adresu (obvodově - **adresa_přerušení**).
- Dojde obvykle k zablokování dalších přerušení

Pak procesor obnoví svoji normální činnost od nastavené adresy, která je vstupním bodem obslužného programu periferie. **Uložení původního obsahu čítače instrukcí je nutné pro návrat na správné místo hlavního programu** (přepisem čítače instrukcí se ztrácí informace o místě, kde byl hlavní program přerušen). **Jsou-li v obslužném programu využívány některé registry**, využívané hlavním programem, potom je potřeba **jejich obsah na začátku obsluhy uložit do zásobníkové paměti a na konci obsluhy opět obnovit včetně stavu příznaků.** Uložení registrů do zásobníku může být zjednodušeno – Existencí dvou bank registrů (Z80, ADSP21xx, atd.), – jednoúrovňovým FIFO základních registrů (TMS320C5x). **Obslužná rutina je ukončena instrukcí RETI nebo RETE** (s povolením přerušení) **a RETI** (se zakázaným přerušením).

Každá periferie napojená na přerušovací systém je obvykle vybavena bitem, který je obvykle při **vzniku žádosti** (**interrupt flag**) nastaven na log.1. Každá žádost je doplněna bitem, který dané přerušení povoluje nebo jej zakazuje (**interrupt mask**). Přerušovací systém je doplněn bitem, který povoluje všechna povolená přerušení nebo zakazuje všechna přerušení. (**Global mask**).

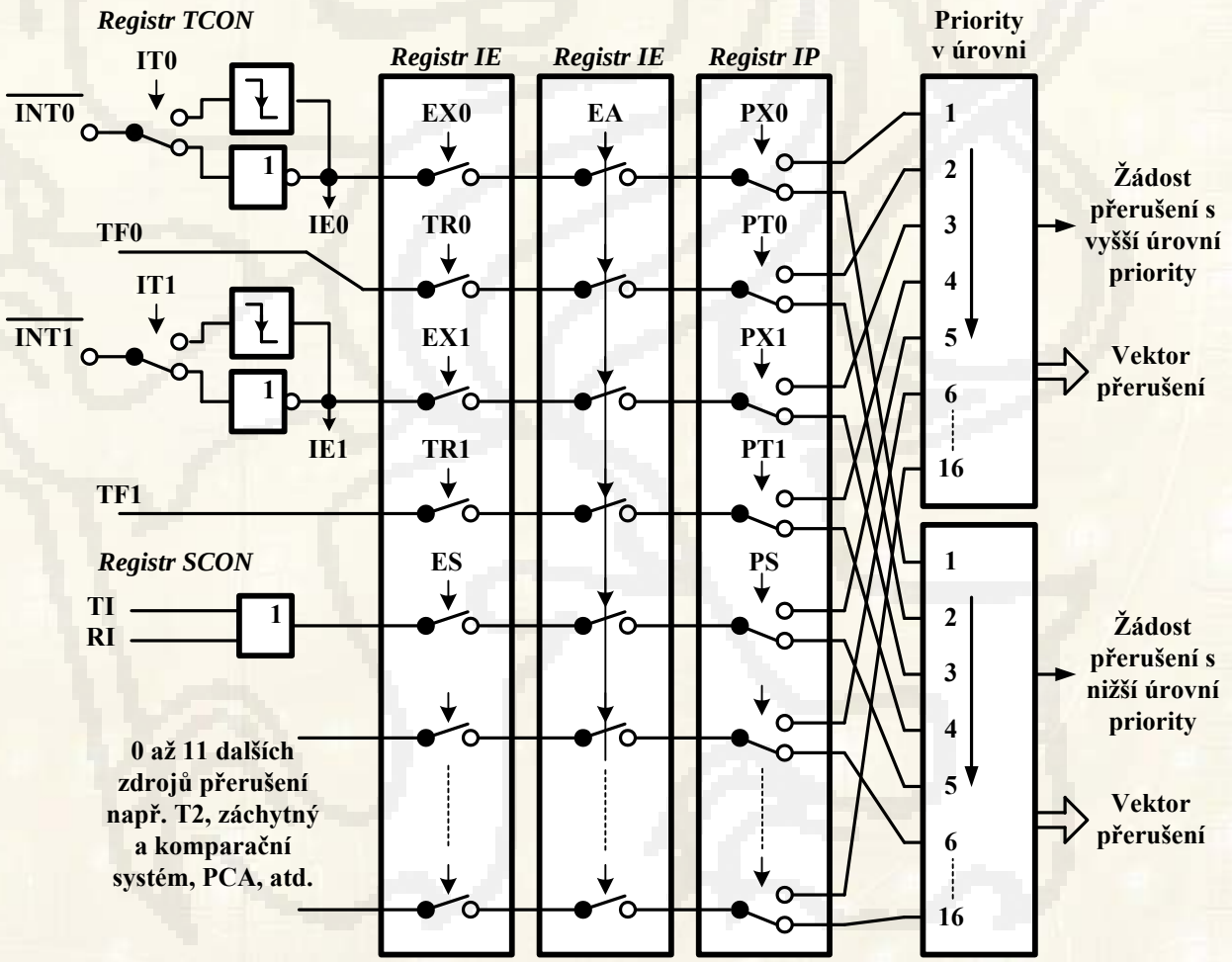
STANDARDNÍ PŘERUŠOVACÍ SYSTÉM

Standardní přerušovací systém je zobrazen na příkladu přerušovacího systému signálového procesoru TMS320C542. Jednotlivé žádosti o přerušení mají, na základě zkušeností z typických aplikací, od výrobce přiřazenu prioritu. Objeví-li se více méně ve stejnou dobu dvě žádosti (jsou vyhodnoceny ve stejnou dobu a obě individuálně i globálně povoleny), pak do zpracování jde nejprve ta z vyšší prioritou. Po dokončení obsluhy je opět provedeno vyhodnocení a přijato přerušení s nejvyšší prioritou. Po návratu z obsluhy přerušení může být vykonána jedna instrukce hlavního programu nebo následovat přímo obsluha dalšího přerušení. U některých procesorů je možné realizovat přerušení v přerušení (nested interrupt). V přerušovací rutině je nastavena nová hodnota IMR a povoleno globální přerušení INTM. Na konci rutiny je třeba zakázat globální přerušení INTM (log.0) a nastavit původní IMR. Každý procesor není vybaven vstupem **nemaskovaného přerušení (NMI)**, který se využívá k řešení havarijních situací (výpadek napájení se zálohováním, atd.). Jsou procesory (např. 8051) u kterých lze z jakéhokoliv přerušení vytvořit přerušení, které se může chovat jako NMI, i když bude ve skutečnosti maskovatelné. Každý procesor též **není vybaven možností předadresováním vektorů přerušení** registrem (viz.IPTR).



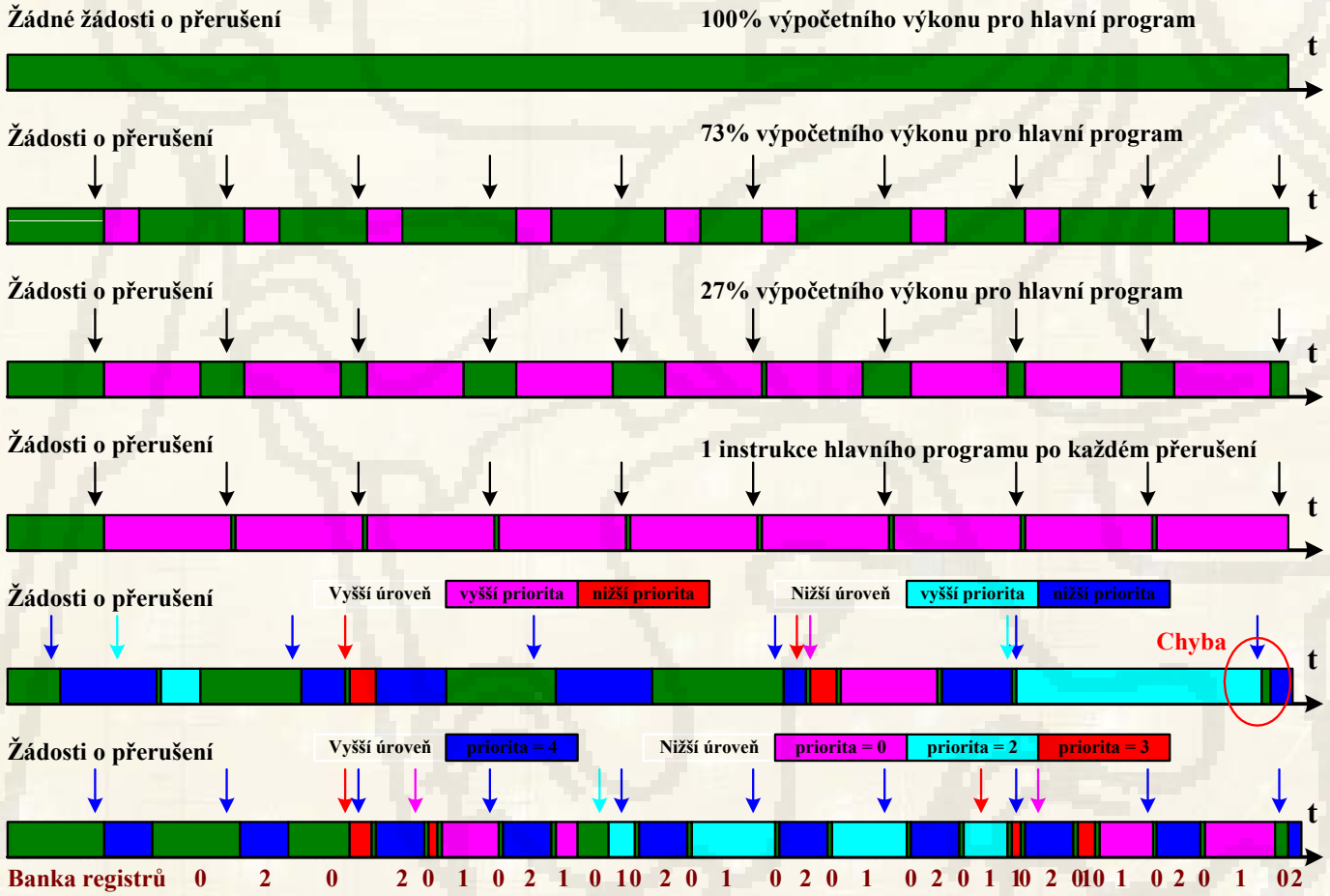
ZÁKLADNÍ PŘERUŠOVACÍ SYSTÉM PROCESORŮ 8051

U procesoru 8051 výrobce pevně přiřadil priority jednotlivým přerušením v každé ze **dvou (dnes i čtyř) úrovní** priority. Nejvyšší prioritu má vnější přerušení (IE0) následované časovačem 0 (TF0), vnějším přerušením (IE1), časovačem 1 (TF1) a sériovým kanálem (RI a TI). Za základními pěti zdroji přerušení následují další, které mají sestupnou prioritu a jejich pořadí není standardizováno. Procesor je vybaven 4-mi bankami registrů R0÷R7, které lze využívat pro přerušení v jednotlivých úrovních bez nutnosti jejich ukládání do zásobníku. Do zásobníku musí být uloženy příznaky (PSW) a střadač (ACC) (případně registr B, DPL DPH). Je-li vyvoláno přerušení v dané úrovni, nemůže být vyvoláno přerušení z dané úrovně byť s vyšší prioritou. Přerušení z **vyšší úrovně priority** může přerušit přerušení s jakoukoliv prioritou z nižší úrovně priority. Úroveň priority se nastavuje v registru IP nebo IP0 a IP1.



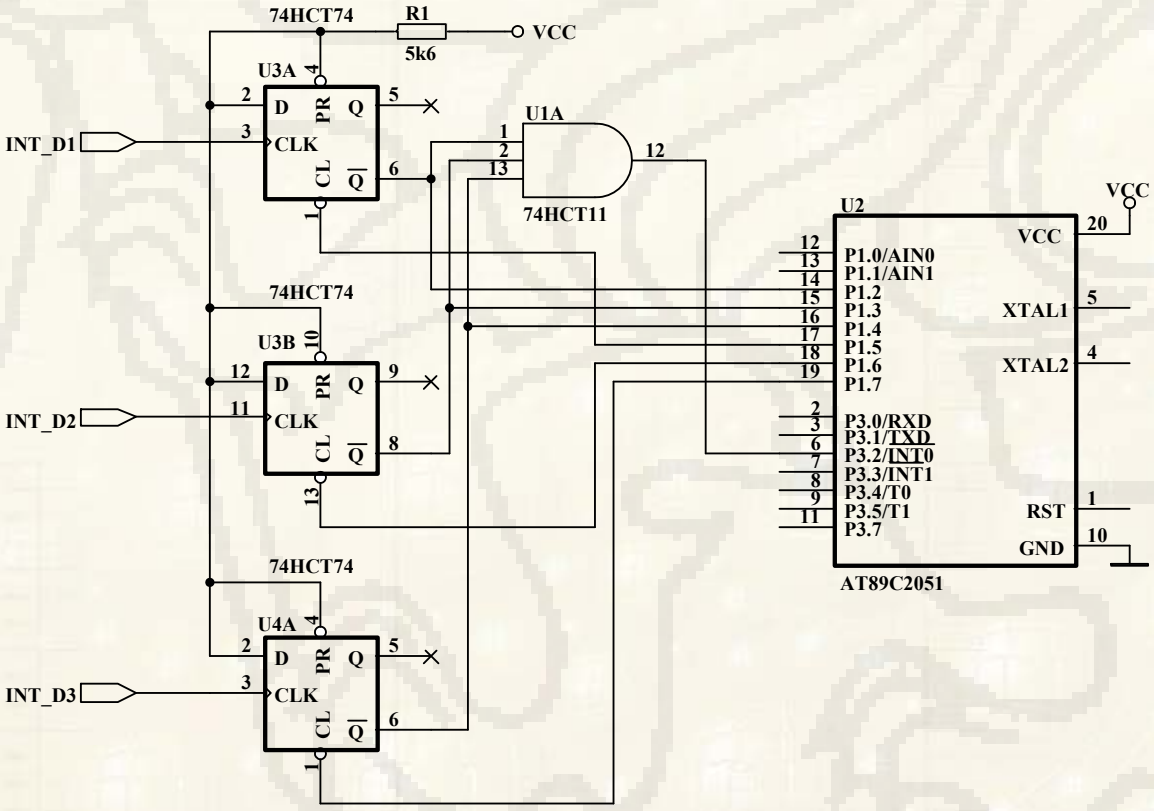
ZATÍŽENÍ PROCESORU PŘERUŠOVACÍM SYSTÉMEM

Přerušovací rutina by měla trvat co nejkratší dobu, zvláště je-li často vyvolávána. **Výrazné snížení výpočetního výkonu pro hlavní program nemusí být na závadu**. Problémy nastávají při dvou a více přerušení viz. příklad pro procesor z řady 8051. Při dvou častých přerušeních může nastat situace viz. **chyba**, při které musíme **zkrátit přerušovací rutiny nebo použít procesor s vyšším výpočetním výkonem.**



ZVĚTŠENÍ POČTU VNĚJŠÍCH PŘERUŠENÍ

Existují aplikace, u kterých potřebujeme sledovat velký počet vnějších žádostí o přerušení. Je relativně málo procesorů s dostatečným počtem vnějších žádostí o přerušení. Další vnější přerušení lze získat u procesorů se záchytným systémem (případně PCA), které nevyužíváme. **Další vnější přerušení můžeme získat obvodovým řešením s využitím jednoho vstupu vnějšího přerušení aktivního v log.0** a několika V/V vývody viz. schéma.



ADRESY PŘERUŠOVACÍHO SYSTÉMU 8051

K využití přerušovacího systému musíme ještě znát, jakým způsobem je realizováno volání obsluhy přerušení. U procesorů 8051 má většina zdrojů přerušení definovanou pevnou adresu, která je volána jako podprogram, po přijetí žádosti o přerušení, dokončení rozpracované instrukce a uložení návratové adresy (adresy instrukce za právě dokončenou instrukcí) do zásobníku. **Zpracování obsluhy přerušení** je zahájeno zápisem odpovídající adresy do čítače instrukcí. Adresy přerušení jsou od sebe vzdáleny pouze o 8 adresových míst a proto na nich obvykle **bývá uložena pouze instrukce nepodmíněného skoku na vlastní obslužný podprogram** zakončený obvykle instrukcí RETI (RETurn from Interrupt). **Obsluha přerušení nesmí být zakončena instrukcí RET.**

Procesor	Klasická 8051		Atmel AT89C51RE2		DS89C430		ADuC834		Adresa obsluhy
Příznak	Zdroj přerušení		Zdroj přerušení	Priorita	Zdroj přerušení	Priorita	Zdroj přerušení	Priorita	
IE0	Vnější přerušení 0		Vnější přerušení 0	1	Vnější přerušení 0	2	Vnější přerušení 0	3	0003h
TF0	Čítač/časovač 0		Čítač/časovač 0	2	Čítač/časovač 0	3	Čítač/časovač 0	5	000Bh
IE1	Vnější přerušení 1		Vnější přerušení 1	3	Vnější přerušení 1	4	Vnější přerušení 1	6	0013h
TF1	Čítač/časovač 1		Čítač/časovač 1	4	Čítač/časovač 1	5	Čítač/časovač 1	7	001Bh
RI+TI	Sériový kanál		Sériový kanál	5	Sériový kanál	6	Sériový kanál	9	0023h
TF2+EXF2			Čítač/časovač 2	6	Čítač/časovač 2	7	Čítač/časovač 2	10	002Bh
CF,CCFn	EPFI	RDYn	Systém PCA	7	Pokles napájení	1	A/D převodník	4	0033h
KBDIT	ES1	ISPI/I2CI	Klávesnice	8	Sériový kanál 1	8	SPI	8	003Bh
TWIIT	EX2	PSMI	TWI	9	Vnější přerušení 2	9	Hlídač napájení	1	0043h
SPIIT	EX3		SPI	10	Vnější přerušení 3	10			004Bh
RI1+TI1	EX4	TII	Sériový kanál 1	11	Vnější přerušení 4	11	Časovač TIME	11	0053h
	EX5	WDS			Vnější přerušení 5	12	Watchdog	2	005Bh
	EWDI				Watchdog	13			0063h
			Nulování procesoru RESET = Nemaskované přerušení s absolutní prioritou						0000h
Přerušení, jejichž příznaky se nulují před vstupem do obsluhy přerušení					Přerušení, jejichž příznak se musí nulovat v obsluze přerušení				

PROGRAMOVÁNÍ PŘERUŠOVACÍHO SYSTÉMU 8051

Typický začátek programu v jazyce symbolických adres

```
RESET:   JMP START          ;Skok na začátek programu
O_INT0:  JMP P_INT0         ;Skok na obsluhu přerušení_0
         ORG 000Bh          ;Nastavení adresy překladu na hodnotu 000Bh
O_CAS0:  JMP P_CAS0         ;Skok na obsluhu časovače_0
         ORG 0013h          ;Nastavení adresy překladu na hodnotu 0013h
O_INT1:  JMP P_INT1         ;Skok na obsluhu vnějšího přerušení_1
         ORG 001Bh          ;Nastav. adresy překladu na hodnotu 001Bh
O_CAS1:  RETI               ;Nepoužívané přerušení
         ORG 0023h          ;Nastav. adresy překladu na hodnotu 0023h
O_SER :  JMP P_SER          ;Skok na obsluhu sériového kanálu
START:   MOV SP,#STACK      ;Nastavení ukazatele zásobníku
HLAVNI:  .......            ;Smyčka hlavního programu
         .......
         LJMP HLAVNI
P_INT0:  .......            ;Obsluha vnějšího přerušení_0
         RETI
P_SER:   JB TI,VYS          ;Obsluha vysílání
         CLR RI             ;Nulování příznaku přijatého znaku
         MOV R1,SBUF        ;Přijatý znak do R1
         RETI               ;Návrat z přerušení
VYS:     CLR TI             ;Nulování příznaku vyslaného znaku
         RETI               ;Návrat z přerušení
```

PROGRAMOVÁNÍ PŘERUŠOVACÍHO SYSTÉMU 8051

Typický začátek programu v jazyce C

```
//   Definice konstant, připojení, globálních proměnných, atd.
main()
{       /* POCATECNI NASTAVENI PRO KONTROLU HLOUBKY ZASOBNIKU */
        for (i=0xFF; i>=0x80; i--) DBYTE[i]=0xAF;
#ifdef  AT89C51ED2
        CKCON0=0x7E;            // Mód X1 jako normální ATMEL
#endif
hlavni: .......                 // Smyčka hlavního programu
        .......
        goto hlavni;
}
//      PODPROGRAMY PRERUSENI
void vnejsi0() interrupt 0 using 3      // Obsluha PC klávesnice
{       ....... ;
        return; }
void time1() interrupt 3 using 1        // Nepoužívané přerušení
{        ;}
void seriovy_kanal() interrupt 4 using 1
{        unsigned char znak;
         if (TI==1) { TI=0; return; }   // Vysilani retezce
         else
         {    RI=0; znak=SBUF; }       // Příjem znaku do proměnné znak
}
```

ATYPICKÉ PROGRAMOVÁNÍ PŘERUŠOVACÍHO SYSTÉMU 8051

Potřebujeme-li v aplikaci přestavět priority přerušení, můžeme přistoupit k programovému řešení priorit, které bylo v historii použito u některých procesorů (např. signálových TI – jedna adresa pro všechna přerušení, jedna adresa pro PCA (8051)). Princip lze snadno uplatnit u přerušení, jejichž **příznak musí uživatel smazat sám v obslužné rutině**. Jsou-li příznaky přerušení mazány automaticky je přestavba priorit poněkud problematická. Výhodnější pravděpodobně bude potřebná přerušení přesunout do vyšších úrovní priority.

Typický začátek programu v jazyce symbolických adres

```
RESET:   JMP START          ;Skok na začátek programu
         ORG 0023h          ;Nastav. adresy překladu na hodnotu 0023h
O_SER :  JMP PRERUS         ;Obsluha sériového kanálu
         ORG 0033h          ;Nastav. adresy překladu na hodnotu 0033h
O_COMP:  JMP PRERUS         ;Obsluha analogového komparátoru
START:   MOV SP,#STACK      ;Nastavení ukazatele zásobníku
HLAVNI:  .....              ;Smyčka hlavního programu
         LJMP HLAVNI
PRERUS:  JB CF, P_CF        ;Skok na obsluhu analogového komparátoru
         JB TI,VYS          ;Druhou nejvyšší prioritu má vysílání znaku
         JB RI,PRIJ         ;Třetí nejvyšší prioritu má příjem znaku
         RETI               ;Návrat z přerušení, Následují jed.obsluhy
VYS:     CLR TI             ;Nulování příznaku vyslaného znaku
         RETI               ;Návrat z přerušení
```

ADRESY PŘERUŠOVACÍHO SYSTÉMU ATmega

U procesorů Atmega má každá periferie spojená s přerušovacím systémem svoji vlastní adresu přerušení (obsluha přerušení není zpožďována) a její příznak se nuluje před vstupem do přerušovací rutiny. Vyjma prvních dvou přerušení je poloha obsluhy přerušení závislá na velikosti procesoru. Priorita přerušení klesá s rostoucí adresou obsluhy.

Procesor				Priorita	Adresa obsluhy
ATmega8	ATmega16	ATmega64	ATmega128		
Zdroj přerušení	Zdroj přerušení	Zdroj přerušení	Zdroj přerušení		
RESET	RESET	RESET	RESET	0	0000h
Vnější přerušení 0	Vnější přerušení 0	Vnější přerušení 0	Vnější přerušení 0	1	0002h
Vnější přerušení 1	Vnější přerušení 1	Vnější přerušení 1	Vnější přerušení 1	2	0004h
TIMER2 COMPare	TIMER2 COMPare	Vnější přerušení 2	Vnější přerušení 2	3	0006h
TIMER2 OVF	TIMER2 OVF	Vnější přerušení 3	Vnější přerušení 3	4	0008h
TIMER1 CAPTure	TIMER1 CAPTure	Vnější přerušení 4	Vnější přerušení 4	5	000Ah
TIMER1 COMP_A	TIMER1 COMP_A	Vnější přerušení 5	Vnější přerušení 5	6	000Ch
TIMER1 COMPB	TIMER1 COMPB	Vnější přerušení 6	Vnější přerušení 6	7	000Eh
TIMER1 OVF	TIMER1 OVF	Vnější přerušení 7	Vnější přerušení 7	8	0010h
TIMER0 OVF	TIMER0 OVF	TIMER2 COMPare	TIMER2 COMPare	9	0012h
SPI, STC	SPI, STC	TIMER2 OVF	TIMER2 OVF	10	0014h
USART, RXC	USART, RXC	TIMER1 CAPTure	TIMER1 CAPTure	11	0016h
USART, UDRE	USART, UDRE	TIMER1 COMP_A	TIMER1 COMP_A	12	0018h
USART, TXC	USART, TXC	TIMER1 COMPB	TIMER1 COMPB	13	001Ah
ADC	ADC	TIMER1 OVF	TIMER1 OVF	14	001Ch
EE_RDY	EE_RDY	TIMER0 COMP	TIMER0 COMP	15	001Eh
ANAlog_COMP	ANAlog_COMP	TIMER0 OVF	TIMER0 OVF	16	0020h
TWI	TWI	SPI, STC	SPI, STC	17	0022h
StoreProgMem_RDY	Vnější přerušení 2	USART0, RXC	USART0, RXC	18	0024h
	TIMER2 COMPare	USART0, UDRE	USART0, UDRE	19	0026h
	StoreProgMem_RDY	USART0, TXC	USART0, TXC	20	0028h
		atd.	atd.	atd.	atd.
		SPM READY	SPM READY	35	0044h

PROGRAMOVÁNÍ PŘERUŠOVACÍHO SYSTÉMU ATmega

Typický začátek programu v jazyce symbolických adres

```
RESET:  JMP START           ;Skok na začátek programu
        rjmp EXT_INT0       ;Skok na obsluhu vnějšího přerušení0
        rjmp EXT_INT1       ;Skok na obsluhu vnějšího přerušení1
        rjmp TIM2_COMP      ;Skok na obsluhu komparace časovače2
        rjmp TIM2_OVF       ;Skok na obsluhu přetečení časovače2
        rjmp TIM1_CAPT      ;Skok na obsluhu zachytného sys. časovače1
        rjmp TIM1_COMPA     ;Skok na obsluhu komparačního sys.časovače2
        atd.

START:   ldi r16,high(RAMEND);Začátek hlavního programu
         out SPH,r16         ;Nastaveni ukazatele zásobníku na vrchol RAM
         ldi r16,low(RAMEND)
         out SPL,r16
         sei                 ;Povolení přerušovacího systému

HLAVNI: .......              ;Smyčka hlavního programu
        .......
        .......
        .......
        LJMP HLAVNI
```

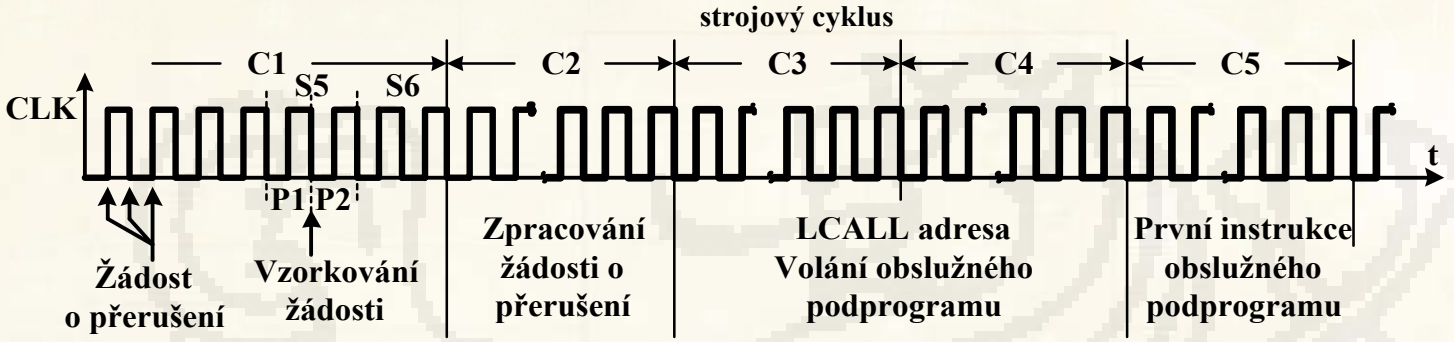
PROGRAMOVÁNÍ PŘERUŠOVACÍHO SYSTÉMU ATmega

Typický začátek programu v jazyce C

```
//   Definice konstant, připojení, globálních proměnných, atd.
main()
{       PORTA=0x08; DDRA=0x00;        // Port A inicializace IIIIOIII
        PORTB=0xA7; DDRB=0x00;        // Port B inicializace OIOIIOOO
        PORTC=0x00; DDRC=0x00;        // Port C inicializace IIIIIIII
        PORTD=0xF2; DDRD=0x00;        // Port D inicializace OOOOIIOI
        TCCR0=0x02;                   // Časovač 0, clock Fs/2=3686,4kHz
        TCNT0=~41;                    // Přednastaveni pro 50us

hlavni: .…….                         // Smyčka hlavního programu
        goto hlavni;
}
//      PODPROGRAMY PRERUSENI
interrupt [EXT_INT0] void vnejsi0()          // Nepoužívané přerušení
{       ; }
interrupt [EXT_INT1] void vnejsi1()          // Obsluha VF čtečky
{       nula=manch1; jedna=manch2; manch1=manch2=0;
        // atd. }
interrupt [UART_RXC] void rxd_prenes()       // Obsluha sériového přijmu
        unsegned char znak;
        if (prijem485==1)                    // Příjem řetězce
        {    znak=UDR; serrxd[pocetrxd]=znak;
             if (pocetrxd<14) pocetrxd++; }
}
```

ODEZVA PŘERUŠOVACÍHO SYSTÉMU PROCESORŮ Z ŘADY 8051



Žádosti o přerušení (příznaky) se vzorkují v době S5P2 každého strojového cyklu procesoru (standardní verze) a **vyhodnocují se v následujícím cyklu**. Jsou-li splněny všechny podmínky pro vyvolání přerušení (neprobíhá přerušení se stejnou úrovní priority, není přerušení zakázané lokálně nebo globálně), potom v cyklech C3 a C4 je generována instrukce LCALL na příslušnou adresu obslužného podprogramu. V cyklu C5 potom může být zpracována první instrukce obslužného podprogramu. Vznikne-li žádost o přerušení s vyšší prioritou před periodou S5P2 strojového cyklu C3, potom bude přerušení obslouženo podle výše uvedených pravidel během strojových cyklů C5 a C6, aniž se vykoná jediná instrukce obslužného podprogramu s nižší prioritou. **Mezi vznikem požadavku na přerušení a první instrukcí obslužného programu proběhne minimálně (3*12+2÷6)/fosc hodinových cyklů**. Odezva na přerušení se prodlouží při platné podmínce blokující přechod do obslužného podprogramu. V případě probíhajícího přerušení se stejnou nebo vyšší úrovní priority, je doba čekání závislá na délce a vlastnostech obslužného podprogramu. **Není-li zpracování žádosti realizováno v posledním cyklu instrukce ⇒ prodloužení odezvy až o 3 strojové cykly (instrukce MUL a DIV trvají 4 cykly).** Probíhá-li právě instrukce RETI nebo instrukce operující s IE nebo IP, prodlouží se doba odezvy nejvíce o 5 strojových cyklů (1 cyklus pro dobíhající instrukci a 4 cykly za dokončení nejdelší následující instrukce MUL nebo DIV). Po instrukci RETI nebo instrukci operující s IE a IP je vykonána ještě jedna instrukce. Využíváme-li v systému pouze jedno přerušení, potom časová odezva bude **maximálně (8*12+2÷6)/fosc hodinových cyklů (vždy delší než 3 a kratší než 9 strojových cyklů). Pozor ⇒ odezva na periodické požadavky fluktuuje v závislosti na zpracovávaných instrukcích ⇒ fázová nestabilita - nelze používat k vzorkování A/D a D/A převodníků, generování signálů, atd.**

ODEZVA PŘERUŠOVACÍHO SYSTÉMU u ATmega a μP s PIPELINE

Odezva na přerušení všech povolených přerušení na procesoru AVR je minimálně čtyři strojové cykly (hodinové cykly). Po čtyřech hodinových cyklech, během nichž je uložen aktuální stav čítače instrukcí do zásobníku, je započato vykonávání odpovídajícího přerušení. Jestliže se žádost o přerušení objeví během více cyklové instrukce, je zřejmé, že musí být nejdříve dokončena a přerušení je pozdrženo. Je-li procesor v módu se sníženou spotřebou sleep mode, potom přerušení je vykonáno se zpožděním o další čtyři hodinové cykly potřebné k opuštění módu sleep. Návrat z přerušovací rutiny trvá čtyři hodinové cykly, během nichž je obnoven stav čítače instrukcí (2byty), ukazatel zásobníku je inkrementován o hodnotu 2 a bit I v registru SREG je nastaven na log.1.

U procesorů s větším rozdělením instrukcí, než na dvě části (ATmega), je při přijatém přerušení potřeba nejprve vyprázdnit všechny rozdělané instrukce a teprve potom je započato načítání prvních instrukcí přerušovací rutiny. V případě návratu z přerušení jsou načtené instrukce za instrukcí RETI (RETE) změněny na instrukce NOP. Aby nedocházelo ke ztrátě strojového času, jsou u některých procesorů zavedeny tzv. **zpožděné instrukce**. S větším počtem překrývaných instrukcí nejen roste doba přístupu do přerušovací rutiny, ale závisí též na umístění zásobníku a nutnosti uložení registrů používaných v příslušném přerušení (banky registrů, interní zásobník, externí umístění zásobníku, jednoúrovňové FIFO strategických registrů).