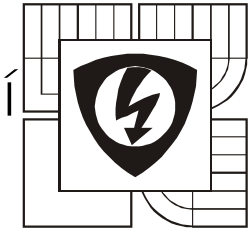
Řadič programu, zřetězené zpracování instrukcí



Kurz: Signálové procesory

Autor: Petr Sysel

Lektor: Petr Sysel

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vytvoření této videopřednášky bylo podpořeno projektem č. CZ.1.07/2.2.00/28.0098
Evropského sociálního fondu a státním rozpočtem České republiky.

Obsah přednášky

Řadič programu

- Stará se o čtení instrukcí, jejich dekódování, správné nastavení řídících signálů, atd.,
- řídí tok programu, podporuje nepodmíněné i podmíněné skoky v programu,
- uchovává informace o aktuálním stavu procesoru – příznaky výsledků aritmetických operací, informace o přerušení, hardwarových cyklech, atd.
- zajišťuje obsluhu přerušení, maskování, návrat z přerušení, atd.

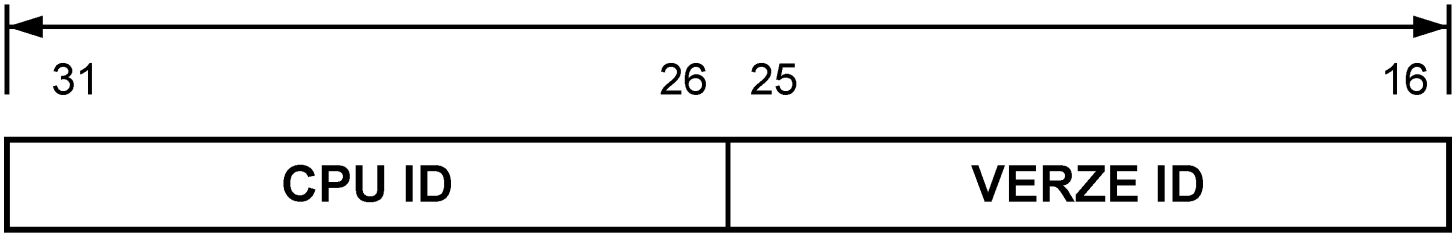
Registry řadiče programu

- `AMR` – Adressing Mode Register,
- `CSR` – Control Status Register,
- `IFR` – Interrupt Flag Register,
- `ISR` – Interrupt Set Register,
- `ICR` – Interrupt Clear Register,
- `ISTP` – Interrupt Service Table Pointer,
- `IRP` – Interrupt Return Pointer,
- `NRP` – Nonmaskable Interrupt Return Pointer,
- `PCE1` – Program Counter E1 phase.

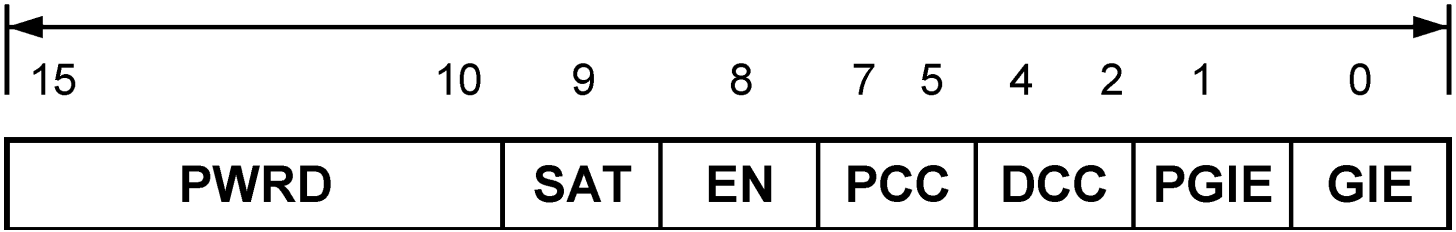
Control Status Register

řídicí stavový registr CSR

identifikace typu a verze procesoru



stavové a řídicí bity



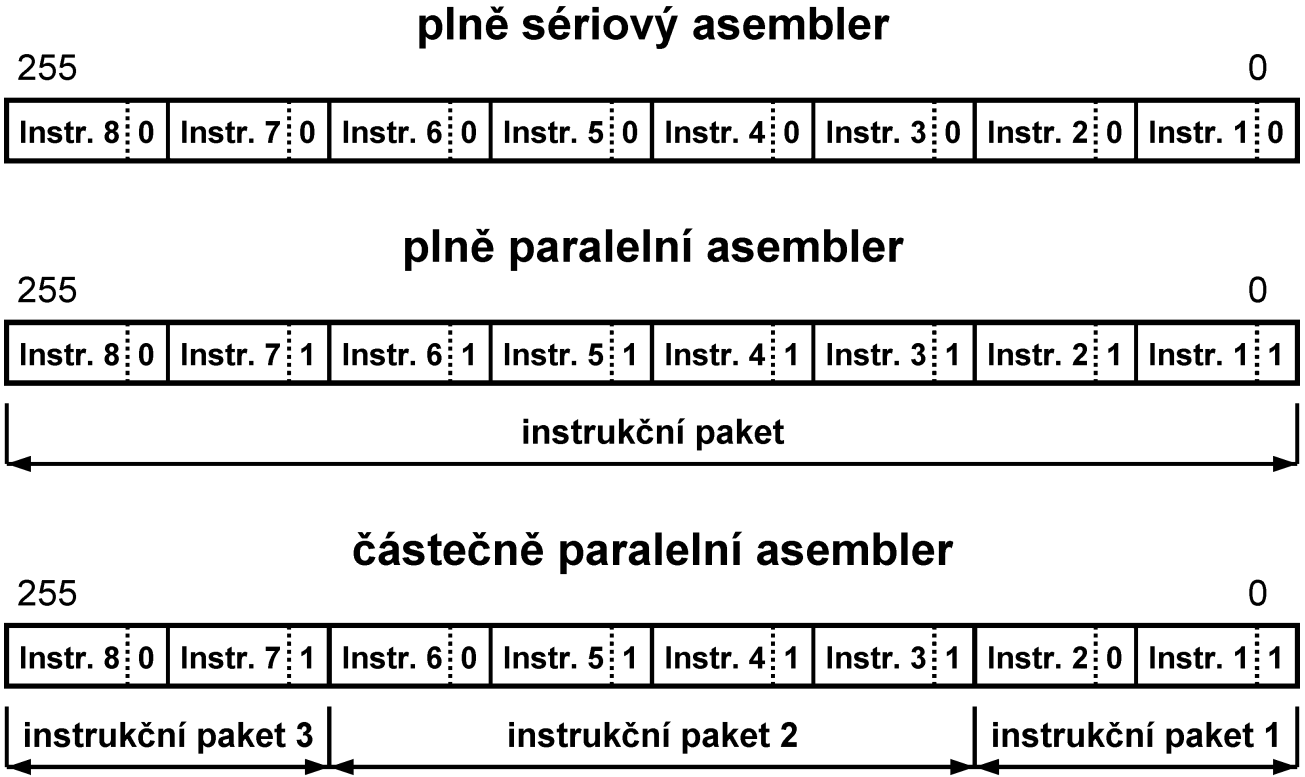
Control Status Register

- `PWRD` – nastavení režimu snížené spotřeby,
- `SAT` – příznak vzniku saturace,
- `EN` – nastavení organizace paměti Little/Big Endian,
- `PCC` Program Cache Control – nastavení použití vnitřní programové paměti,
- `DCC` Data Cache Control – nastavení použití vnitřní datové paměti,
- `PGIE` Previous General Interrupt Enable – předchozí hodnota povolení přerušení,
- `GIE` Global Interrupt Enable – globální povolení přerušení.

Čtení a dekódování instrukce

- Instrukce jsou 32bitové skládané po 8 do instrukčního paketu,
- registr Instruction Latch, ze kterého probíhá dekódování instrukcí, je 256bitový,
- rozlišení instrukcí, které patří k jednomu instrukčnímu paketu se děje bitem p v instrukčním slově,
- z výše uvedeného mimo jiné plyne, že instrukční paket musí být vždy zarovnán na násobky 32 (256bitovou adresu).

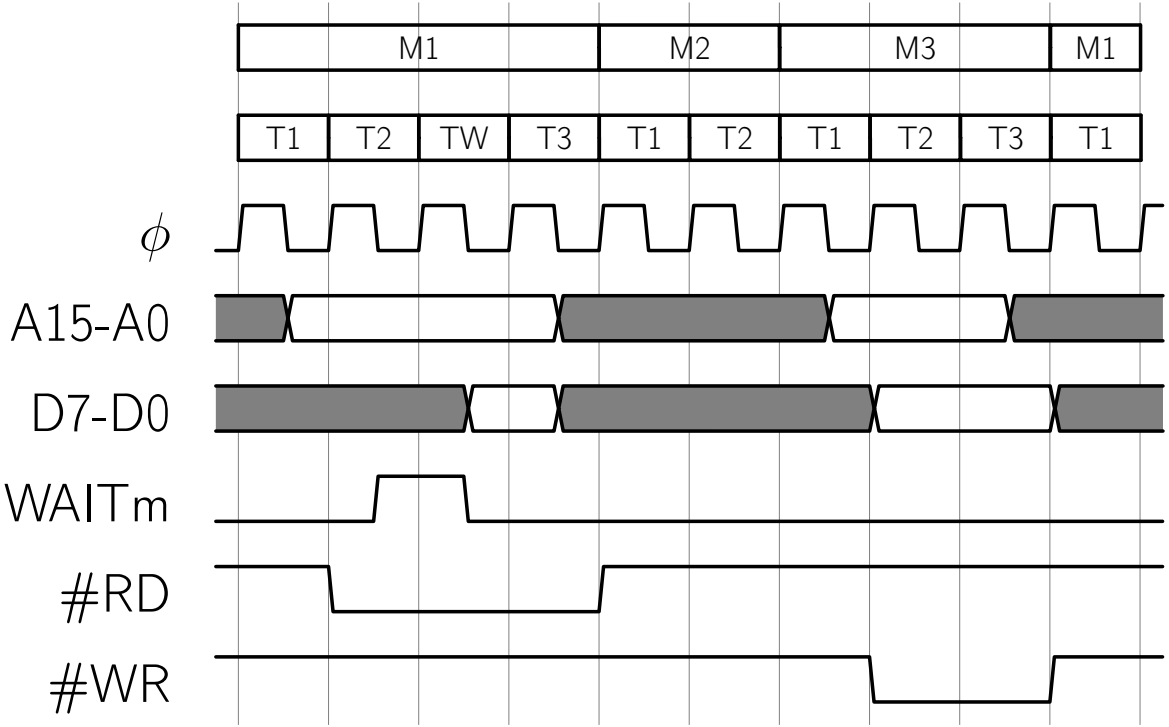
Instrukční paket



Časový popis činnosti procesoru

- Výkon každé instrukce je rozložen do strojových cyklů `M` – machine cycle,
- každý strojový cyklus může být tvořen jedním nebo několika takty hodinového signálu `T` – clock cycle,
- většinou čím výkonější mikroprocesor, tím více strojových cyklů je v jedné instrukci,
- strojový cyklus reprezentuje sekvence činností, které jsou opakovány v různých instrukcích – např. generaci adresy, zápis do paměti, apod.,
- během strojového cyklu jsou generovány řídící signály, které určují typ komunikace (čtení, zápis) na řídící sběrnici,
- v některých případech jsou během strojového cyklu generovány také stavové signály, které blíže specifikují strojový cyklus,
- první strojový cyklus každé instrukce tvoří tzv. fetch cycle, který provádí čtení instrukce z paměti do řadiče programu.

Příklad časového popisu činnosti



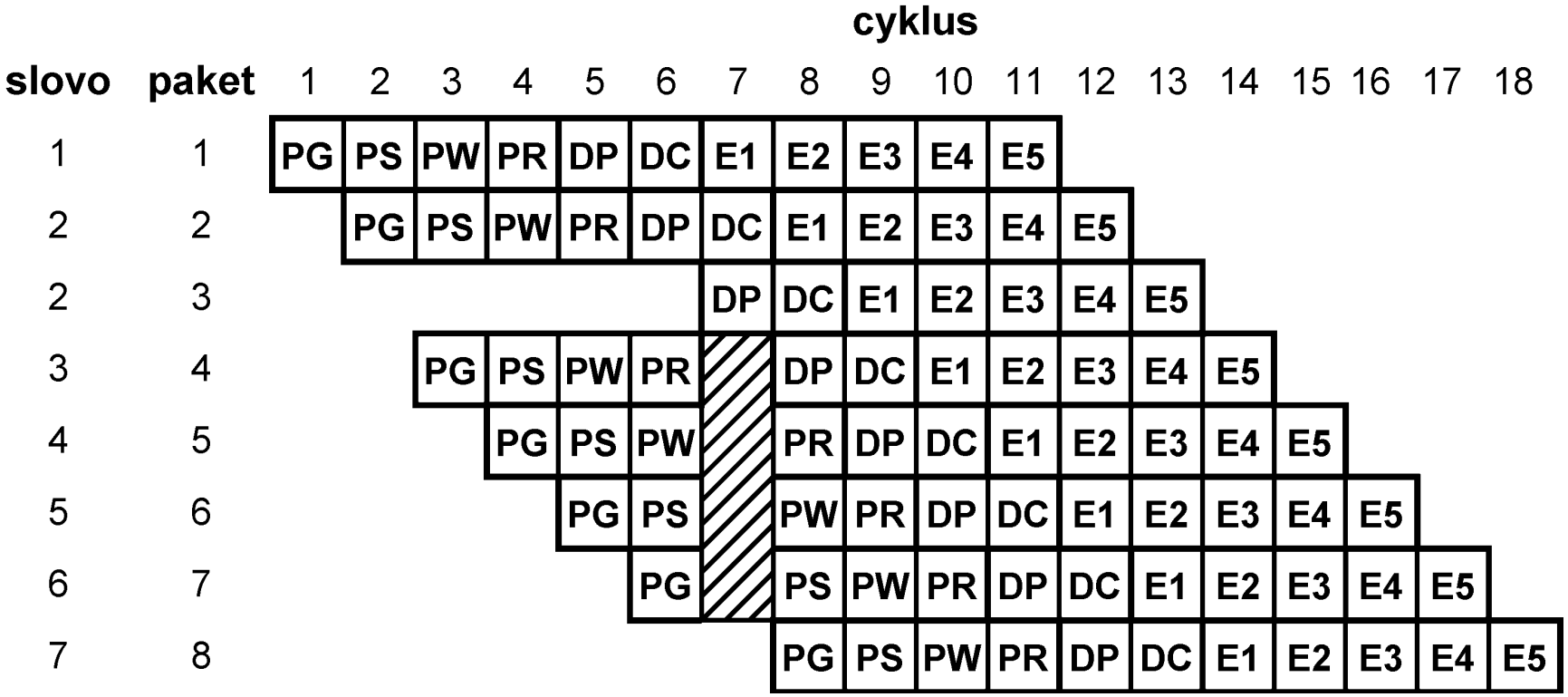
Časový popis u procesoru TMS320C6416

- První fáze Fetch je rozdělena do 4 hodinových cyklů
  - PG – Program address Generate,
  - PS – Program address Send,
  - PW – Program access ready Wait,
  - PR – Program fetch packet Receive.
- Druhá fáze Decode je rozdělena do 2 hodinových cyklů
  - DP – instruction DisPatch,
  - DC – instruction DeCode.
- Třetí poslední fáze Execution je rozdělena až do 5 hodinových cyklů E1, E2, E3, E4 a E5 podle složitosti instrukce.

Zřetězené zpracování instrukcí – pipelining

- V jednom okamžiku je možné provádět různé instrukční cykly různých instrukcí,
- instrukce jsou zpracovávány paralelně, což vede ke zvýšení výkonu,
- obecně čím výkonější procesor, tím větší počet fází (vrstev) zřetězeného zpracování instrukcí podporuje (TMS320C6000 má 11 vrstvé zřetězené zpracování instrukcí),
- používá se především u procesorů zaměřených na výkon, u procesorů pro řídící aplikace, kde se často využívají skoky a podmíněné skoky je využití problematické.

zřetězené zpracování instrukcí



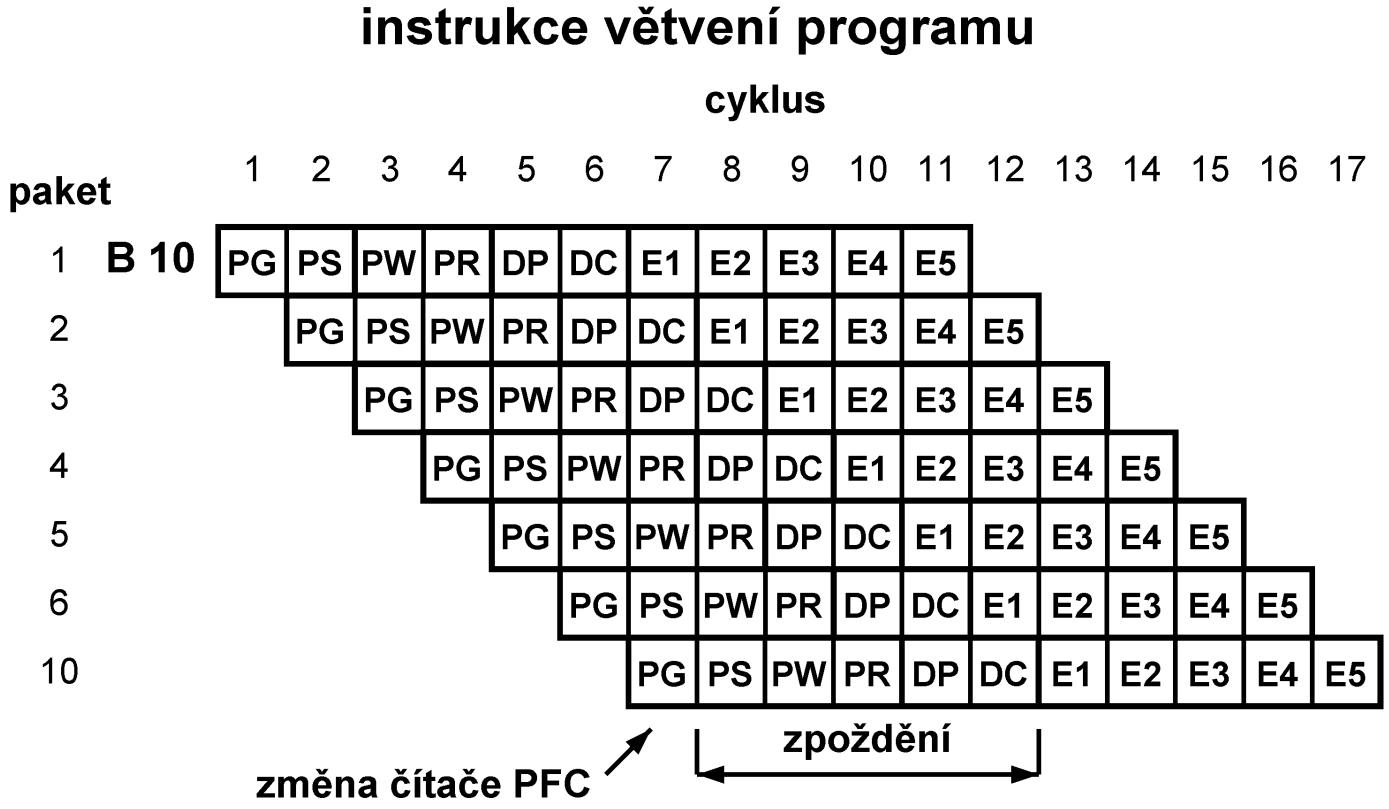
Problémy zřetězeného zpracování instrukcí

- Konflikt v datech – vstupní operandy instrukce jsou výstupní hodnoty předchozí instrukce nebo instrukce čte hodnoty ukládané předchozí instrukcí do paměti,
- řešení konfliktu v datech je vložení čekacích cyklů (stall cycle) buď automaticky za běhu nebo vložení instrukcí `nop` během překladu,
- narušení pipeliningu při instrukcích skoků – v případě skoku je nutné provést první fáze zřetězeného zpracování instrukcí na nové adrese,
- nejhorší jsou podmíněné skoky – není možné dopředu rozhodnout,
- řešení problémů se skoky je použití tzv. zpožděných skoků (delayed) – instrukce jejichž zpracování bylo zahájeno jsou dokončeny,
- za instrukcí skoku musí být uveden patřičný počet platných instrukcí, které budou ještě vykonány, pokud nechci vykonávat instrukce ve zpoždění, musí zde být instrukce `nop`.

Zpoždění jednotlivých instrukcí

Instruction Type	Delay Slots	Unit Latency	Read Cycles	Write Cycles	Branch Taken
NOP	0	1			
Store	0	1	i	i	
Single cycle	0	1	i	i	
Two Cycle	1	1	i	i+1	
Multiply (16×16)	1	1	i	i+1	
Four cycle	3	1	i	i+3	
Load	4	1	i	i,i+4	
Branch	5	1	i		i+5

Příklad větvení programu



Časový popis u procesoru DSP56858

- Pre-Fetch 1 `P1` – adresa na adresovou sběrnici paměti programu `PAB`,
- Pre-Fetch 2 `P2` – řídící signály pro čtení z paměti, adresa je dekódována a je započato čtení,
- Instruction Fetch `IF` – čtení z paměti programu, instrukce je uložena do registru `instruction latch`,
- Instruction Decode `ID` – instrukce je dekódována,
- Address Generation `AG` – adresy dat jsou na adresové sběrnice `XAB1` a `XAB2`,
- Operand Pre-Fetch 2 `OP2` – řídící signály pro čtení/zápis z paměti, adresa dat je dekódována a je započato čtení z paměti dat,
- Execute and Operand Fetch `EX` – jsou přečtena data z paměti nebo zapsána na do paměti, jsou vykonány jednoduché aritmetické instrukce nebo započato zpracování složitějších instrukcí,
- Execute 2 `EX2` – složitější aritmetické instrukce jsou dokončeny a výsledky jsou zapsány do cílových registrů

Příklad provádění instrukcí

- Příklad instrukcí

```
move.w  X:(r0),a     ; n1 -- 1 slovo, 1 instrukční cyklus
add     a,b          ; n2 -- 1 slovo, 1 instrukční cyklus
move.w  b,c          ; n3 -- 1 slovo, 1 instrukční cyklus
move.w  c1,X:$0C00   ; n4 -- 2 slova, 2 instrukční cykly
inc.w   c            ; n5 -- 1 slovo, 1 instrukční cyklus
```

- jednotlivé fáze provedení instrukcí

	instrukční cyklus											
vrstva	1	2	3	4	5	6	7	8	9	10	11	12
P1	n1	n2	n3	n4	n4e	n5	·	·	·	·	·	·
P2		n1	n2	n3	n4	n4e	n5	·	·	·	·	·
IF			n1	n2	n3	n4	n4e	n5	·	·	·	·
ID				move	add	move	move	move	inc	·	·	·
AG					move	add	move	move	move	inc	·	·
OP2						move	add	move	move	move	inc	·
EX							move	add	move	move	move	inc
EX2								—	—	—	—	—

Příklad vložení čekacích cyklů

- Příklad instrukcí

```
nop                 ; n1 -- normal state
add     x0,a        ; n2 -- execute phase
mpy     x0,y1,b     ; n3 -- execute phase
move.w  b,X:(r0)+   ; n4 -- 1 stall cycle
                    ;  (závislá na předešlé instrukci)
mac     x0,y0,a     ; n5 -- ex and ex2
mac     x0,y0,a     ; n6 -- ex and ex2
sub     y1,a        ; n7 -- late execution
                    ;  (závislá na předešlé instukci)
bne     LABEL       ; n8 -- 1 stall cycle
                    ;  (závislá na předešlé instrukci)
```

Příklad vložení čekacích cyklů

- Jednotlivé fáze provedení instrukcí

	vrstva zřetězeného zpracování instrukcí							
cyklus	P1	P2	IF	ID	AG	OP2	EX	EX2
1	n1							
2	n2	n1						
3	n3	n2	n1					
4	n4	n3	n2	nop				
5	n5	n4	n3	add	—			
6	n6	n5	n4	mpy	add	—		
7	n7	n6	n5	—	mpy	add	—	
8	—	—	—	move	—	mpy	add	—
9	n8	n7	n6	mac	move	—	mpy	add
10	n9	n8	n7	mac	mac	move	—	mpy
11	n10	n9	n8	sub	mac	mac	move	—
12	—	—	—	—	sub	mac	mac	—
13	n11	n10	n9	bne	—	sub	mac	mac
14	n12	n11	n10		bne	—	—	mac
15	n13	n12	n11			bne	—	sub
16	n14	n13	n12				bne	—

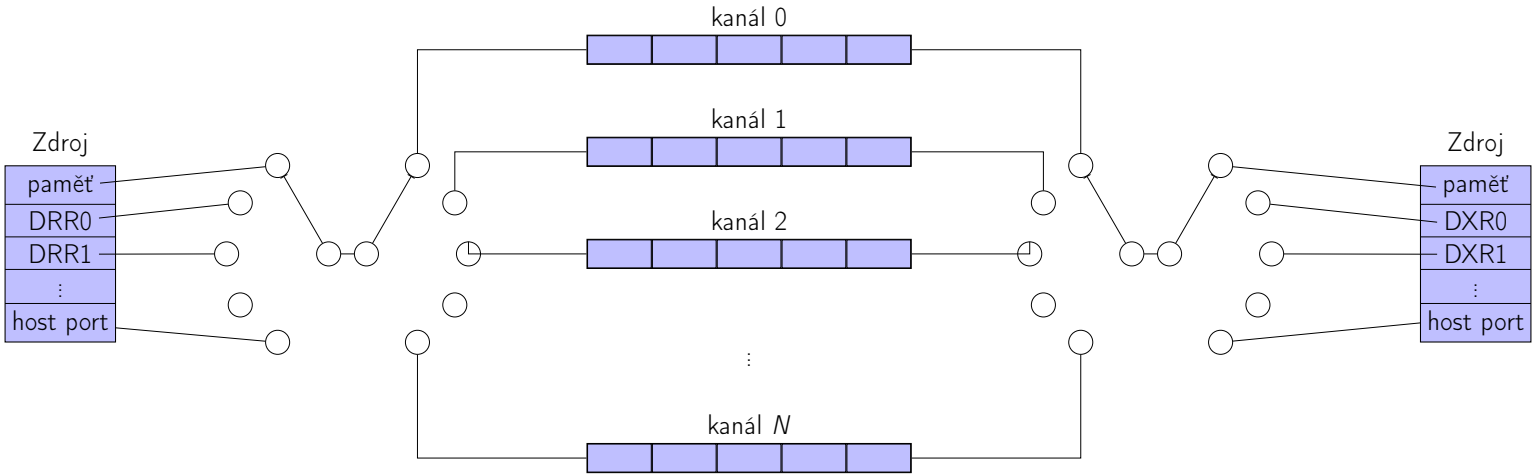
Řadič DMA

- Slouží pro přenos dat mezi vnitřní pamětí, vnější pamětí, periferiemi na čipu.
- Přenos je uskutečňován bez asistence jádra signálového procesoru.
- Přenos lze uskutečňovat také mezi HPI (Host Port Interface) a vnitřní nebo vnější pamětí.
- Přenos je synchronizován událostmi.
- Řadič může generovat přerušení.

Kanály řadiče DMA

- Přenos je uskutečňován pomocí kanálů.
- Každý kanál čte hodnoty z jednoho portů do vyrovnávací paměti FIFO (First In First Out) – `source`.
- Hodnoty z vyrovnávací paměti zapisuje do stejného nebo jiného portu – `destination`.
- Celkem může být nakonfigurováno šest kanálů DMA0-DMA5, které provádí činnost současně.
- Každý kanál má dvě sady řídicích registrů, které řídí jeho činnost – pracovní a konfigurační.

Blokové schéma řadiče DMA



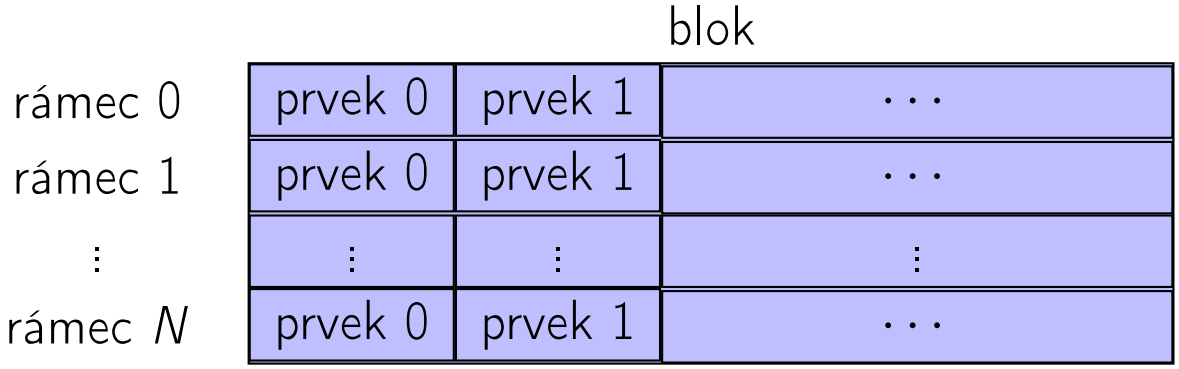
Činnost řadiče DMA

- Každý port řadiče cyklicky kontroluje kanál DMA0, DMA1, DMA2, DMA3, DMA4, DMA5, HPI, DMA0, DMA1, ...
- Pokud daný kanál vyžaduje na daném portu obsluhu (čtení nebo zápis), port ji provede.
- Často lze každému kanálu lze nízkou nebo vysokou prioritu – kanály s nízkou prioritou pak obslouženy až pokud nevznikne žádný požadavek s vysokou prioritou.
- Pokud je některý kanál zakázán, je zakázán na všech portech a jeho požadavky se nekontrolují.

Činnost řadiče DMA

- Přenos probíhá po blocích – block.
- Každý blok je složen z určitého počtu rámců – frame.
- Rámec je složen z určitého počtu prvků – element.
- Prvek představuje 8-, 16- nebo 32bitovou hodnotu, která je přenesena najednou – nemůže být přerušena jiným kanálem.

Rozdělení přenosu na rámce a elementy



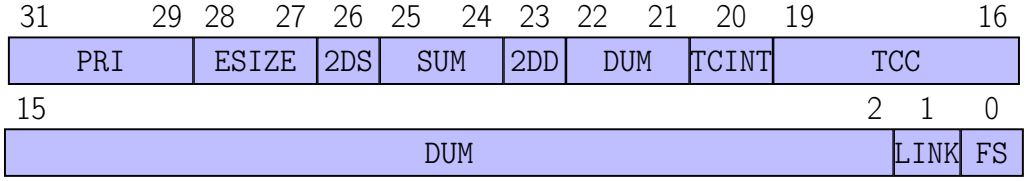
Činnost řadiče DMA

- Po skončení přenosu prvku může dojít ke změně adresy zdroje i cíle.
- Adresa může být:
  - Konstantní – nedojde ke změně,
  - automatické povýšení po přenosu – adresa se automaticky zvýší o velikost prvku,
  - automatické s jedním indexem – po přenosu prvku se adresa zvýší o hodnotu indexu,
  - automatické se dvěmi indexy – po přenosu prvku se adresa automaticky zvýší o hodnotu indexu prvků. Pokud to však byl poslední prvek v rámci, adresa se zvýší o hodnotu indexu rámce.
- Indexy mohou být i záporné, musí však splňovat podmínku zarovnání na velikost dat.

Deskriptor DMA kanálu

	31 Deskriptor 0		
byte 0	EDMA Channel Options Parameter (OPT)		OPT
byte 4	EDMA Channel Source Address (SRC)		SRC
byte 8	Array/frame count (FRMCNT)	Element count (ELECNT)	CNT
byte 12	EDMA Channel Destination Address (DST)		DST
byte 16	Array/frame index (FRMIDX)	Element index(ELEIDX)	IDX
byte 20	Element count reload (ELERLD)	Link address (LINK)	RLD

Význam bitů OPT deskriptoru.



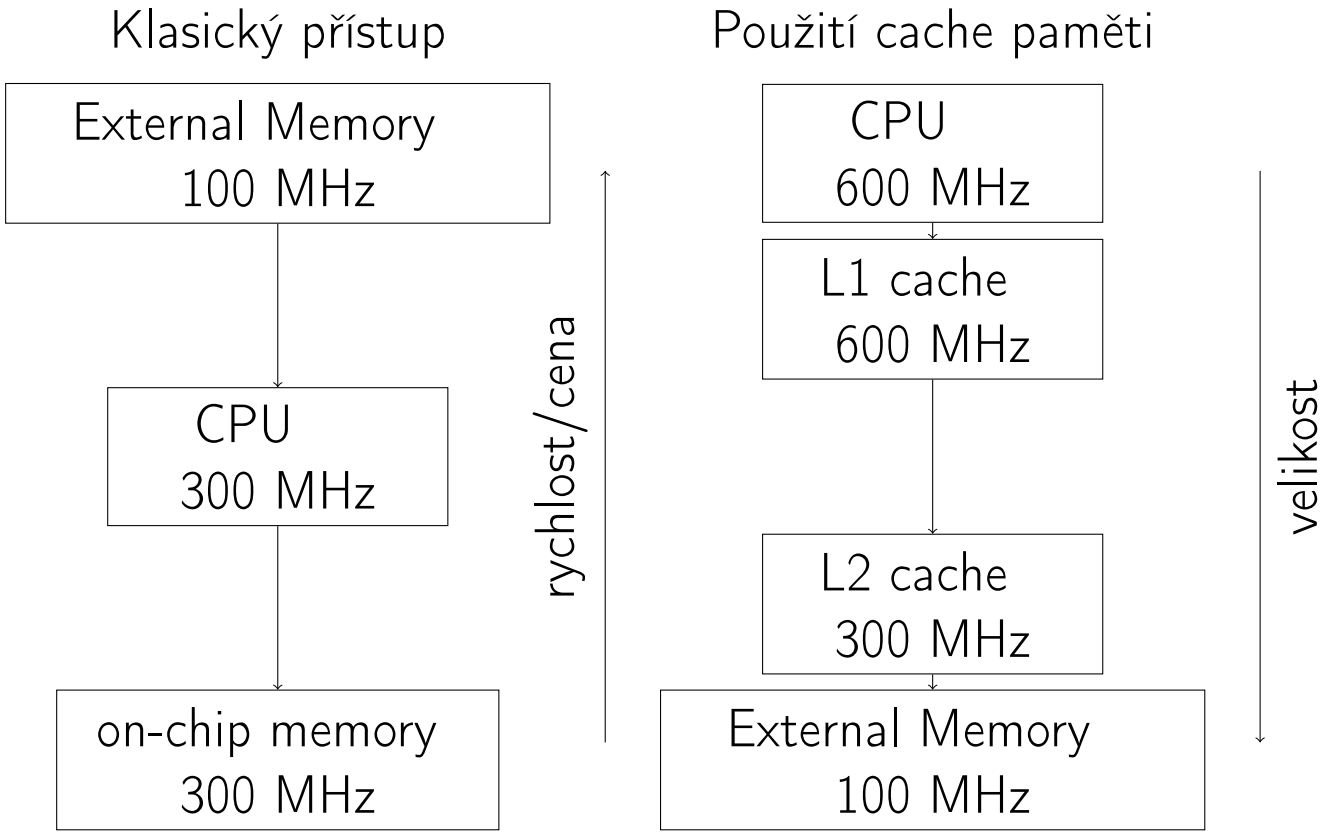
Činnost řadiče DMA

- Přenos každého kanálu může být synchronizován událostí od periferií nebo signálem na pinu procesoru.
- Synchronizován může být přenos jednoho prvku nebo celého rámce.
- Synchronizováno může být čtení nebo zápis.
- Pokud synchronizační signál přijde před tím, než kanál dokončí předchozí operaci, je signál ignorován.

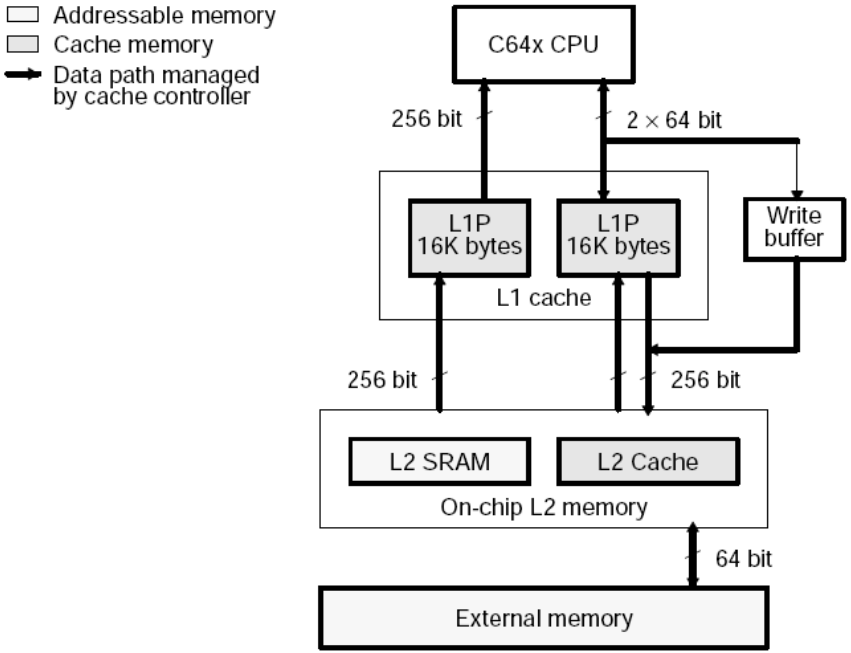
Činnost řadiče DMA

- Každý kanál DMA může generovat přerušení pro jádro signálového procesoru.
- Lze zvolit událost, která způsobí generování přerušení:
  - Dokončení přenosu bloku,
  - začátek přenosu posledního rámce v bloku,
  - dokončení přenosu rámce,
  - dokončení přenosu první poloviny aktuálního rámce,
  - zmeškání synchronizačního impulsu,
  - vypršení času pro přístup do paměti.

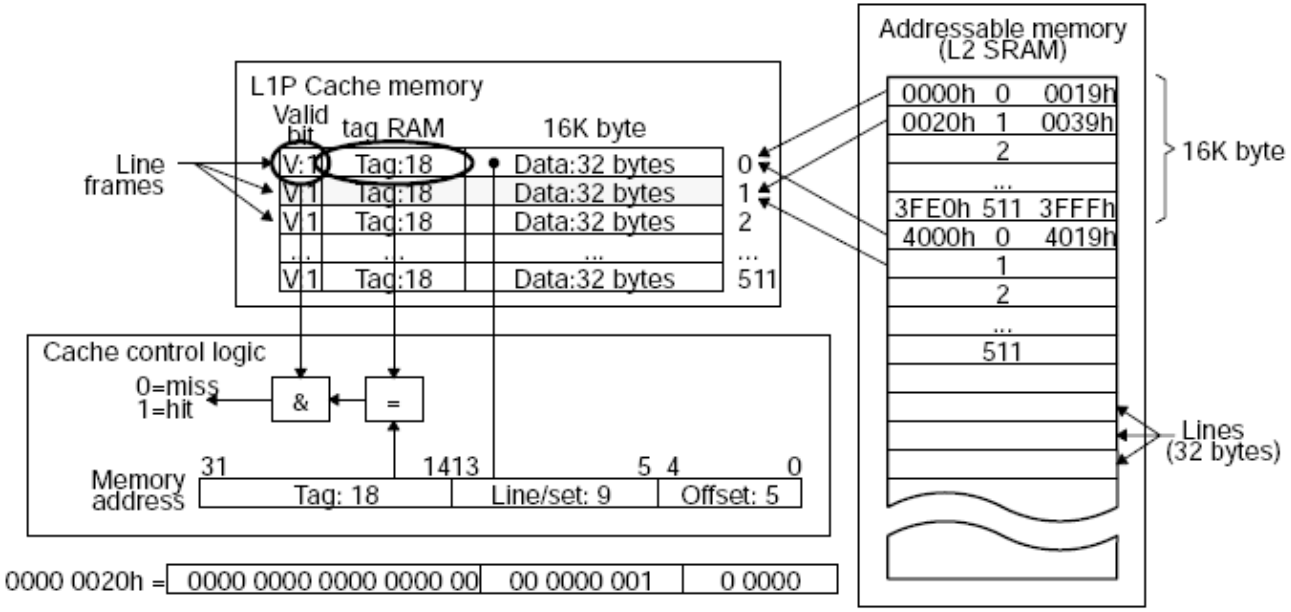
Důvody zavedení paměti CACHE



Paměť cache u procesorů TMS320C64xx



Direct-mapped cache



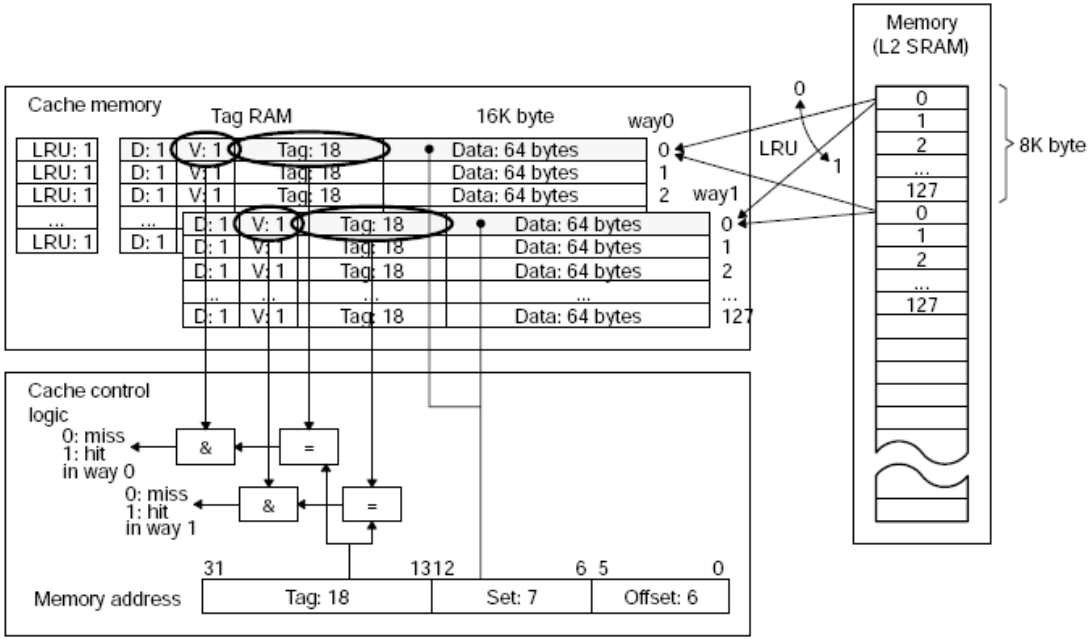
Možné operace

- Compulsory Misses – vynucené minutí
  - Na začátku je paměť cache prázdná,
  - při prvním čtení z nějaké adresy se data do paměti cache přesune celý jeden řádek,
  - nastaví se odpovídající část adresy (tag) a bit platnosti řádkku (valid),
  - tomuto minutí se nedá zabránit.
- Read Hits – čtení z cache paměti
  - Při dalším čtení se porovná část tag adresy s daným řádkem,
  - pokud tag souhlasí a řádek je označen jako platný, proběhne čtení z cache paměti.

Možné operace

- Read Misses – Capacity Misses
  - Postupným načítáním dojde k obsazení všech řádků cache paměti (vyčerpání kapacity),
  - s dalším čtením se začnou řádky cache paměti přepisovat,
  - řešením je zmenšit velikost programu nebo zvětšit paměť cache.
- Read Misses – Conflict Misses
  - Nedojde k vyčerpání kapacity, ale např. proběhne skok na adresu, která je mapována na stejný řádek paměti cache,
  - při čtení z paměti je obsah řádku cache paměti zahozen a nahrazen obsahem na nové adrese,
  - při skoku zpět se musí původní obsah znovu načíst z vnější paměti,
  - řešení je vhodné zarovnání/uspořádání dat.

2way set-associative cache



Možné operace

- Compulsory Misses
  Read Hits
  Read Misses – Conflict Misses
  Read Misses – Capacity Misses
  - Fungují podobně jako v předchozím případě, s tím že jsou k dispozici dvě sady řádků paměti cache,
  - v případě, že je řádek už obsazen, provede se zápis duálního řádku paměti cache,
  - podle posledního čtení se nastaví bit LRU – Least Recently Used,
  - pokud dojde ke konfliktu, nahradí se řádek, ke kterému nebylo naposledy přistupováno.

Možné operace

Protože je tato cache paměť použita pro data, je možné do ní provádět i zápis:

- Write Misses
  - Alokace řádku v paměti cache se provádí pouze při čtení (read-allocate),
  - pokud dojde k zápisu na adresu, která není alokována v paměti cache, provede se zápis přímo do vyšší paměti,
  - pro vyrovnání případné pomalejší reakce vnější paměti je použit jednoduchý bufer.
- Write Hits
  - Zápis do alokovaného řádku se provádí v cache paměti (write-back),
  - v případě, že se provádí zápis na adresu alokované v paměti cache, provede se zápis do odpovídajícího řádku,
  - přitom se neprovádí zápis do vyšší paměti, pouze se nastaví bit D – dirty – k označení, že došlo ke změně řádku,
  - zápis do vyšší paměti se provede až v okamžiku Conflict Misses nebo Capacity Misses.

Koherence paměti cache

- Cache paměť funguje spolehlivě, pokud k adresám alokovaným v paměti cache nepřistupuje nikdo jiný,
- v okamžiku, kdy se např. DMA kanál provede zápis na takovouto adresu, v paměti cache zůstává pořád předchozí hodnota (incoherent),
- v případě DMA řadiče funguje automatický mechanismus (snoop) pro synchronizaci pamětí (coherence),
- v okamžiku, kdy se zjistí takovýto přístup (čtení nebo zápis), provede se případný write-back, řádek cache paměti se zneplatní a provede se DMA přístup,
- v případě, že k podobný přístup realizuje někdo z vnějšku, je nutné to zajistit programově funkcemi `CACHE_invL2`, `CACHE_wbL2`, `CACHE_wbInvL2`.