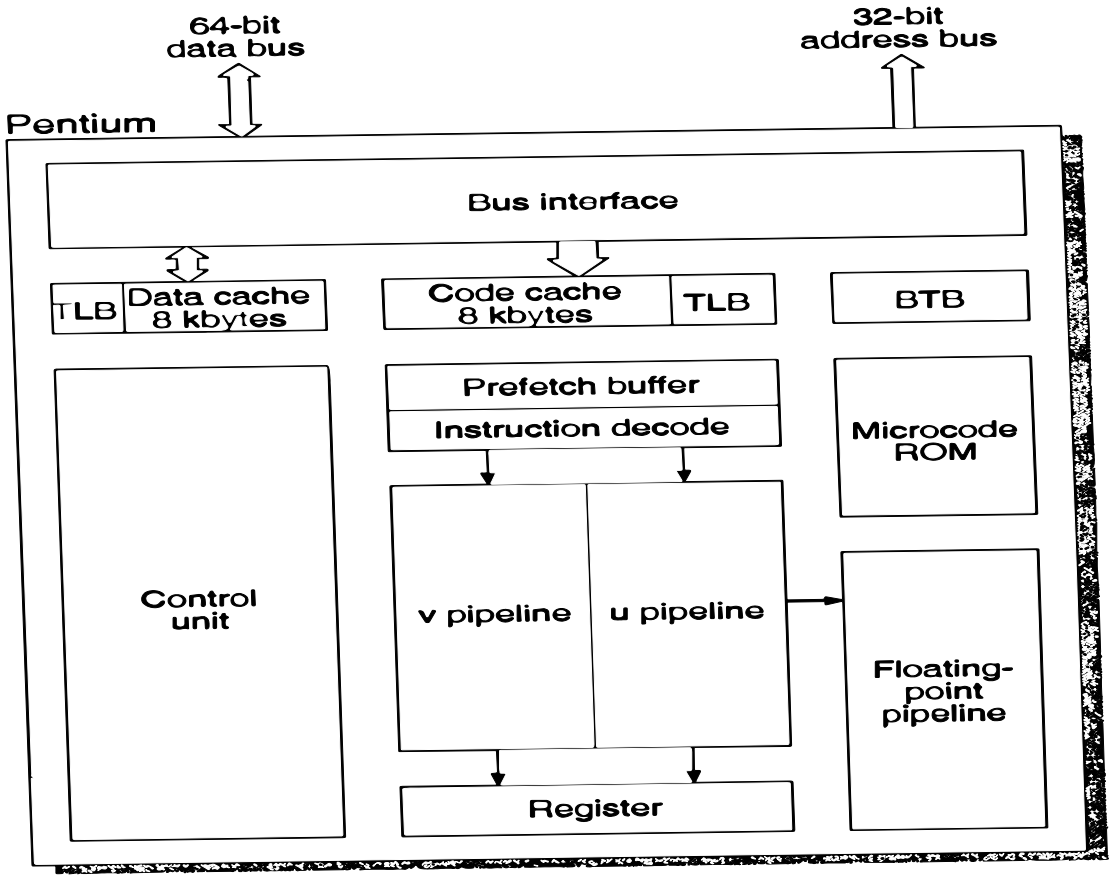
CHARAKTERISTIKA PROCESORU PENTIUM

První verze:

- Verze Pentia 200 Mhz – uvádělo se 330 MIPS (srovnávalo se s 54 MIPS procesoru 486DX2-66).

Struktura Pentia

- Rozhraní – 64 bitů datová sběrnice, 32 bitů adresová sběrnice.



Obr. 1 Blokové schéma prvních typů Pentií;;;

- **Bus interface** (řízení sběrnice) – komunikace s vnějším světem (data, adresa, řídicí signály).
- Rozdělení rychlé vyrovnávací paměti na dvě – **8 kB rychlá vyrovnávací paměť pro data** (data cache), **8 kB rychlá vyrovnávací paměť pro kód** (code cache) – odlišnost od předcházejících typů procesorů.
- Každá rychlá vyrovnávací paměť má svou **TLB** (**T**ranslation **L**ookaside **B**uffer), v níž jsou uchovány posledně používané lineární adresy a k nim odpovídající adresy fyzické.
- Napájecí napětí je 3.3 V.
- První typy byly dodávány v pouzdře PGA (Pin Grid Array) s 273 vývody, později v pouzdře SPGA (Staggered Pin Grid Array, stagerred – uspořádaný střídavě) s 296 vývody a na svém čipu o rozměrech 12,8 x 12,8 mm integruje 3,1 milionu tranzistorů.
- Pentium je prvním **superskalárním procesorem firmy Intel** - více než jedna fronta pro zřetězené zpracování instrukcí.
- Pentium má dvě takovéto fronty označované jako **u**, **v**. Tato vlastnost umožňuje procesoru tzv. **superskalární zpracování instrukcí**.
- Možnost dokončit až dvě instrukce zároveň (v každé frontě jednu) - první typy procesorů Pentium

dosahovaly při stejné frekvenci vyššího výkonu než procesory 80486.

- Zásady pro superskalární zpracování:
    1. Následující instrukce nesmí být závislá na instrukci předcházející (následující instrukce nesmí potřebovat výsledek instrukce předcházející).
    2. Obě instrukce musí být jednoduché, tj. nejsou prováděny mikroprogramově, ale hardwarově.
- Pro tyto účely jsou vypracována pro mikroprocesory Intel tzv. **párovací pravidla** (pairing rules), která specifikují, které instrukce mohou být **párovány**, tzn. zařazeny do front **u**, **v** tak, že splňují podmínky pro paralelní zpracování.
- Každá instrukce zůstane v každém stupni pouze po dobu jednoho cyklu.

Párovací pravidla pro zařzazování instrukcí do front u, v

- **Pravidlo 1**
    - Obě instrukce v páru musí být tzv. **jednoduché**.
    - Jednoduché instrukce – jsou realizovány hardwarově (tzn. nikoliv mikroprogramově).

- **Příklady** (aritmetické nebo logické instrukce):

MOV reg,reg/mem/imm

MOV mem,reg/imm

ALU reg,reg/mem/imm

ALU mem,reg/imm

INC reg/mem

DEC reg/mem

POP reg/mem

LEA reg,mem

JMP/Jcc NEAR/CALL

NOP

Poznámka: všimněme si, že i instrukce operující s pamětí jsou považovány za jednoduché a dejme si to do relace s informací o architekturách RISC, kde je silná snaha o omezení komunikace s pamětí při realizaci instrukce (operandy aritmeticko-logických operací jsou uloženy v paměti).

- **Pravidlo 2**
  - Instrukce se do obou front **u, v** zavádějí současně (do fronty **u** instrukce s nižším pořadím).

- Proto: instrukce nepodmíněného skoku, podmíněného skoku a volání funkcí mohou být zařazovány pouze do fronty **v.**
- Získá se tak čas na předpovězení, jak bude provádění programu ovlivněno těmito instrukcemi skoku (kdyby se zaváděly do fronty **u**, pak se následující instrukce zavede do fronty **v**, a přitom se nebude možná na základě výsledku skoku provádět).
- Pokud se zavede instrukce podmíněného skoku do **v**, pak instrukce za ní následující není ještě zavedena do vstupní fronty, o tom se bude teprve rozhodovat předpovězením dalšího průběhu.

- **Pravidlo 3**

  Instrukce posuvu (*SAL/SAR/SHL/SHR reg/mem,1* a *SAL/SAR/SHL/SHR reg/mem,imm*) a rotace (*RCL/RCR/ROL/ROR reg/mem,1*) mohou být zařazeny pouze do fronty **u**.

  Zdůvodnění (domněnka): pro realizaci těchto činností je vybavena pouze fronta **u**.

- **Pravidlo 4**

  Operandem obou instrukcí nesmí být stejný registr (explicitní nebo implicitní).

Příklad (explicitní registr):

MOV eax,0001h

ADD ecx,eax

Důsledek: instrukce *ADD* ve frontě **v** se musí pozdržet, dokud není zajištěno, že instrukce *MOV* ve frontě **u** zapsala do *eax* hodnotu 0001h.

Příklad (implicitní nebo nepřímo adresovaný registr):

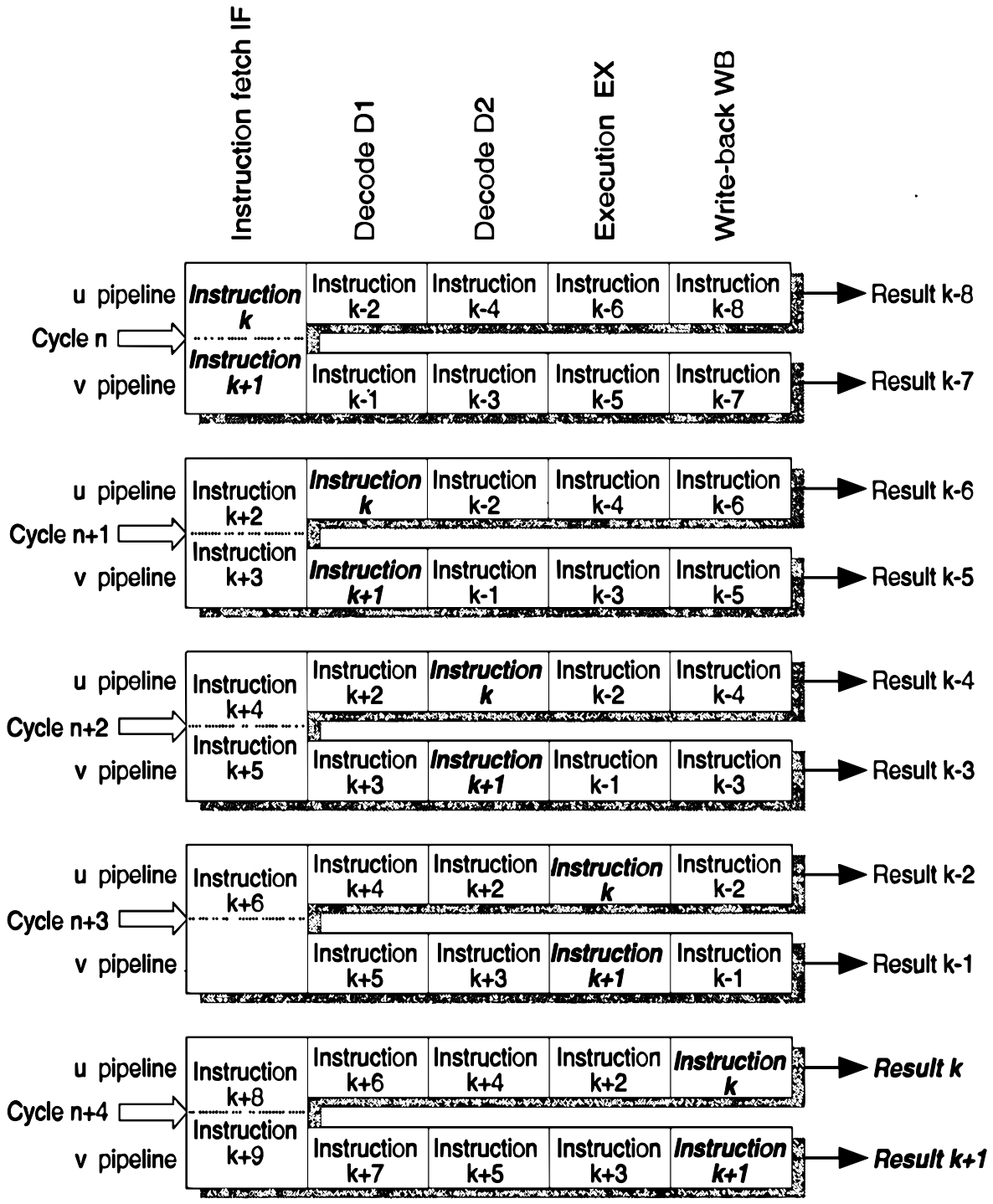
Příklad 1: Instrukce *IMUL mem32* ukládá 64 bitový výsledek násobení *eax*mem32* do dvojice implicitních registrů *edx:eax* => pokud by byla taková instrukce zařazena do fronty **u**, pak by nemohla být do fronty **v** zařazena jiná instrukce operující s registry *edx*, *eax*.

Příklad 2: Instrukce zařazené do fronty **u**, které operují s příznaky v registru *EFlag* – např. instrukce *CMP* - srovnej (carry, overflow, parity, sign, zero, … jsou nastavovány podle výsledku srovnání) - nemohou být párovány např. s instrukcemi *ADC* (add with carry) nebo *SBB* (subtract with borrow) ve frontě **v**, poněvadž bity registru *EFlag* jsou při těchto instrukcích modifikovány.

- **Pravidlo 5**

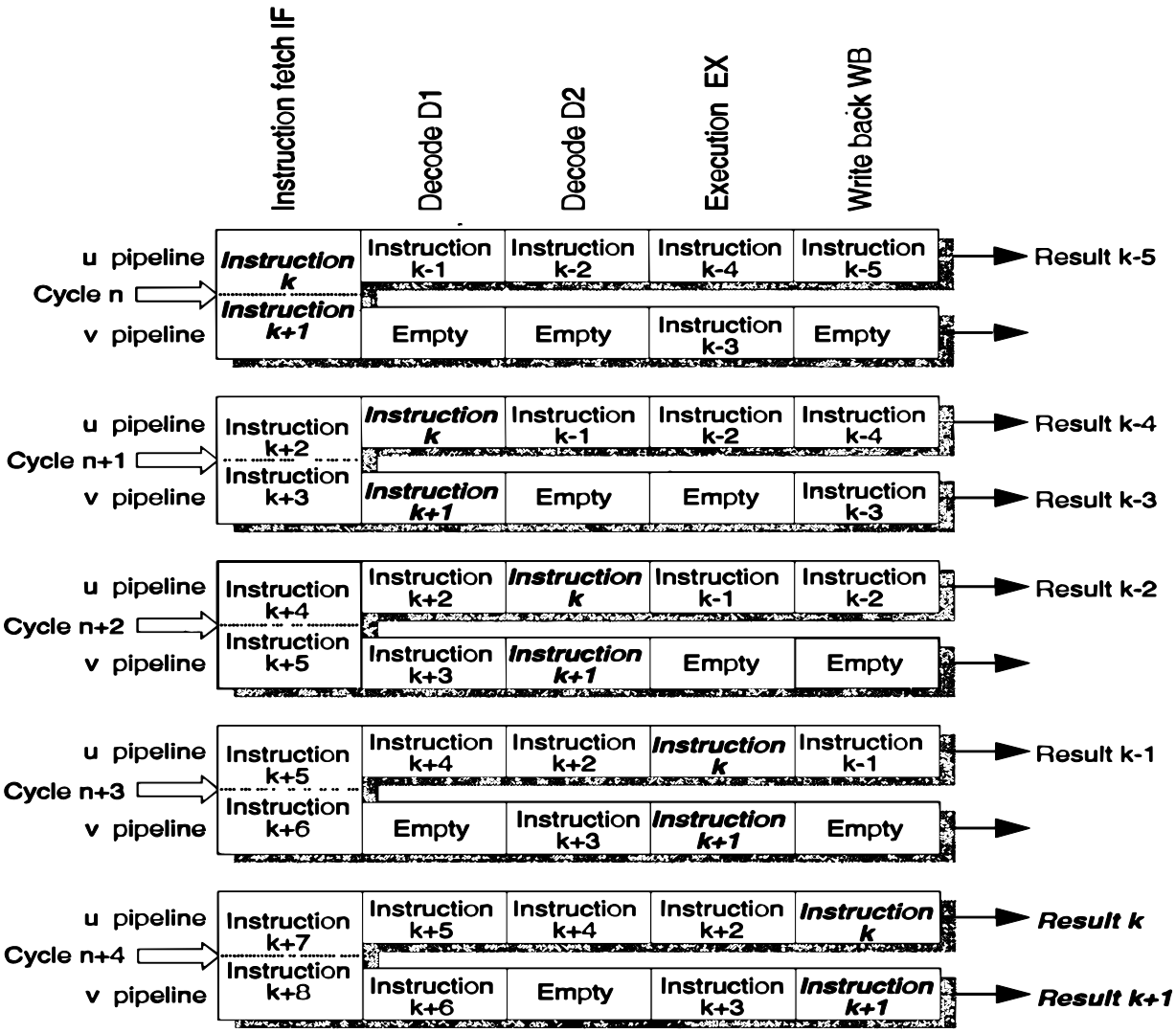
  Instrukce, které obsahují prefix, mohou být zařazeny pouze do fronty **u**.

Příklady párování instrukcí



Obr. 3 Pětistupňová struktura obou front pro zpracování instrukcí s operandy typu integer

- Na obr. 3 je zobrazena situace, kdy párování je možné vždy.



Obr. 4 Příklad řazení instrukcí do front **u**, **v**

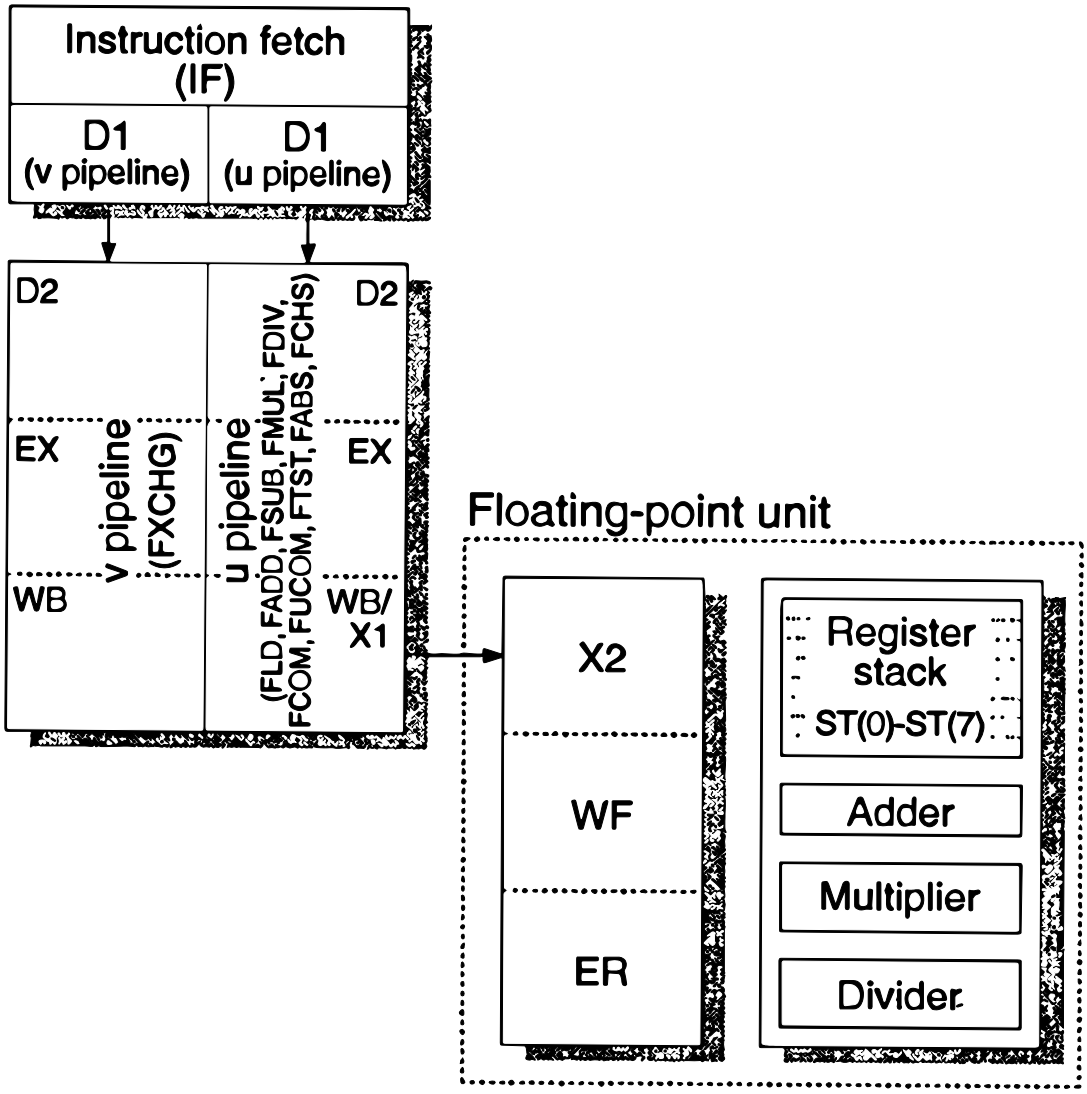
- Obr. 4 – párování instrukcí není vždy realizovatelné:

  Párovány mohou být instrukce k-4 s k-3, k s k+1, k+2 s k+3, k+5 s k+6.

Samostatně musí být prováděny instrukce k-5, k-2, k-1, k+4.

Jednotka FPU

- Kromě prostředků pro zpracování instrukcí pro práci s operandy typu integer, existuje v Pentiu jednotka **FPU** (nahrazující koprocesor předcházejících typů mikroprocesorů), v níž jsou realizovány instrukce zpracovávající operandy v pohyblivé řádové čárce.
- Zařazení jednotky **FPU** – provádění instrukce pak sestává z více (např. 8) stupňů.
- Před jednotkou **FPU** jsou instrukce řazeny do dvou front **u**, **v**, tyto fronty jsou totožné s frontami pro provádění instrukcí s operandy typu integer.
- Na jednotku **FPU** je možné pohlížet jako na komponentu integrovanou do procesoru (na rozdíl od architektury procesor-koprocesor v I80486).
- Další novum – sčítačka, dělička a násobička jsou realizovány hardwarově.



Obr. 5 Struktura jednotky FPU v Pentiu

- Důsledek: první typy Pentií byly 2 – 5 x rychlejší než mikroprocesory I80486DX2.
- Obě fronty instrukcí **u**, **v** sestávaly z pěti stupňů podílejících se na realizaci instrukce:

  Instruction Fetch (IF),

  Decoding 1 (D1),

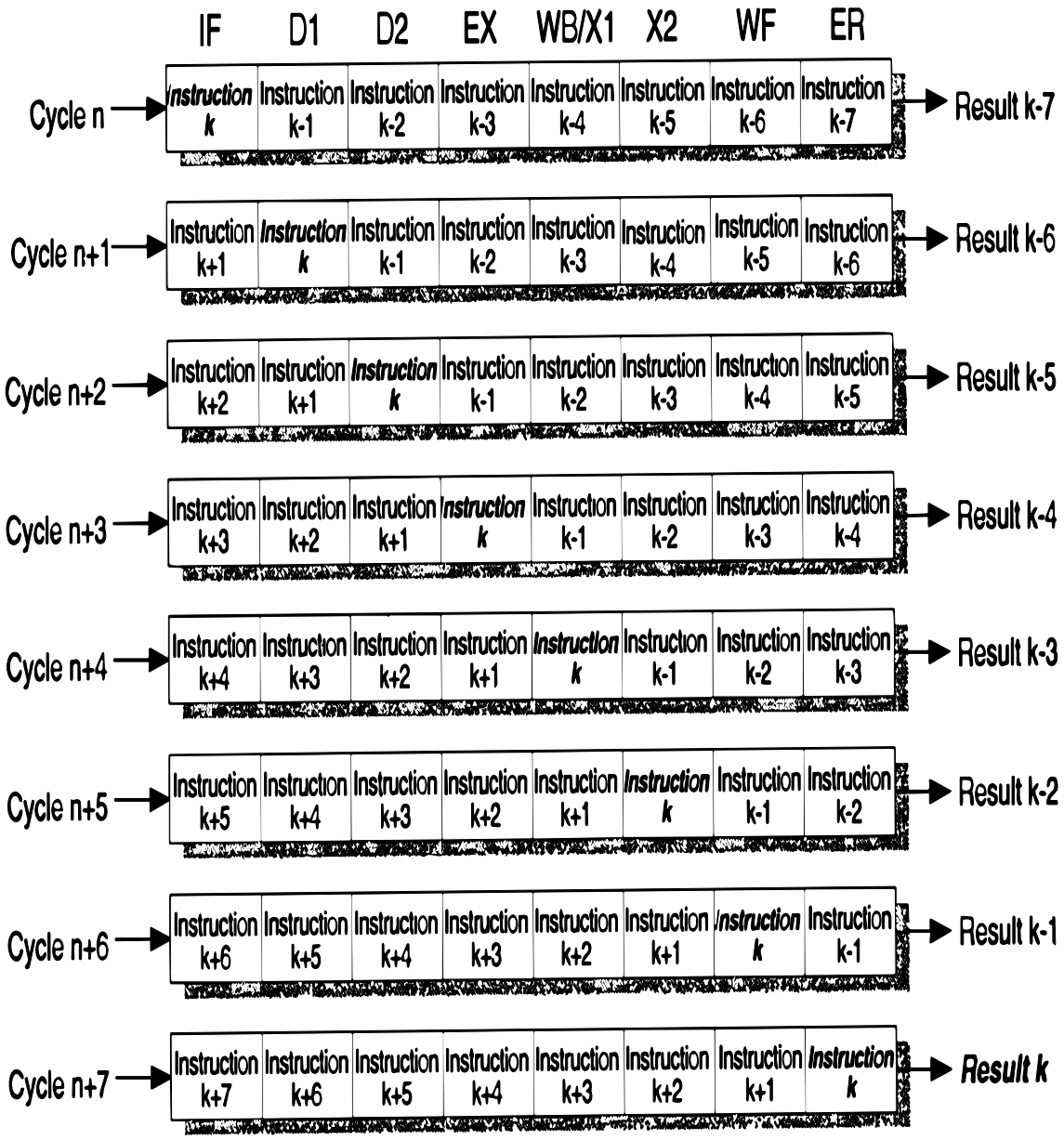
  Decoding 2 (D2),

Execution (EX), obsahuje ALU,

Write Back (WB) – přístupy do rychlé vyrovnávací paměti

- Sdružování instrukcí do dvojic prováděných současně je možné pouze tehdy, pokud jsou splněna určitá dodatečná párovací pravidla.
- **Párovací pravidlo**:

  Instrukce operující s operandy typu integer nesmí být párována s instrukcí operující s operandy v pohyblivé řádové čárce.



Obr. 6 Posloupnost provádění instrukcí v 8 stupňové frontě FPU v Pentiu

Jednotka Instruction Fetch (IF)

- Načítá z rychlé vyrovnávací paměti (cache hit) nebo z operační paměti (cache miss) 2 instrukce, každou z nich do vyrovnávací paměti velikosti 32 B.

- Pokud přečtenou instrukcí je instrukce jiná než skoková (podmíněný nebo nepodmíněný skok), předá jednotka IF instrukci do dekodéru D1.
- Pokud je právě přečtenou instrukcí instrukce skoková, pak jednotka IF zahájí spolupráci s jednotkou **BTB** (Branch Target Buffer) – výsledkem této komunikace je předpověď, jestli nastane skok či nikoliv.
- Stav, kdy je předpovězeno, že nastane skok, se označuje jako *taken branch* (**pozor:** jde pouze o předpověď, definitivně se o skoku – přechodu na jinou instrukci rozhodne, až se bude instrukce provádět).
- Stav *taken branch* (skoková instrukce byla identifikována ve frontě **u**) – instrukční fronta **u** se zablokuje, do instrukční fronty **v** se začne načítat z místa, kam ukazuje skoková instrukce.
- Pokud se později zjistí (v jednotce EX), že predikce nebyla správná, pokračuje se v provádění instrukcí z fronty **u**.
- Znamená to, že technika BTB předpokládá predikci a **na základě predikce čtení instrukcí z místa**, kam ukazuje skok.

Jednotka D1 (Decoding 1)

- Obsahuje dva paralelně pracující dekodéry.
- Rozhoduje se zde, zda je možné instrukce *k* a *k+1* označit za párové a zda je možné je provádět paralelně ve frontách **u** a **v**.
- Pokud ano, pak se instrukce *k* zapíše do fronty **u** a instrukce *k+1* do fronty **v**.
- Obě instrukce pak procházejí frontami paralelně.
- Jednotka Decoding 1 pracuje podle **pravidel pro párování**.
- Pokud není možné instrukce *k* a *k+1* zpracovávat paralelně (není splněno některé z párovacích pravidel), zařadí se do fronty **u** instrukce *k*, načte se instrukce *k+2* a posoudí se, zda je možné paralelně zpracovávat instrukce *k+1* a *k+2*.

Jednotka D2 (Decoding 2)

- Dekóduje adresy operandů a čte je.

Jednotka WB (Write Back)

- Zápis výsledku (týká se pouze instrukce s operandy typu integer).

Jednotka X1

- Je to vlastně první stupeň jednotky **FPU.**
- Jednotka provádí konverzi operandů.
- Identifikuje, zda prováděné instrukce jsou tzv. bezpečné (*safe instructions*) – kontroluje, zda nedojde např. k přetečení.

Jednotka X2

- Provádí vlastní operaci s čísly zobrazenými v pohyblivé řádové čárce.
- Provedení této operace je časově výrazně náročnější než pro celá čísla (integer).

Jednotka WF (FP register write stage)

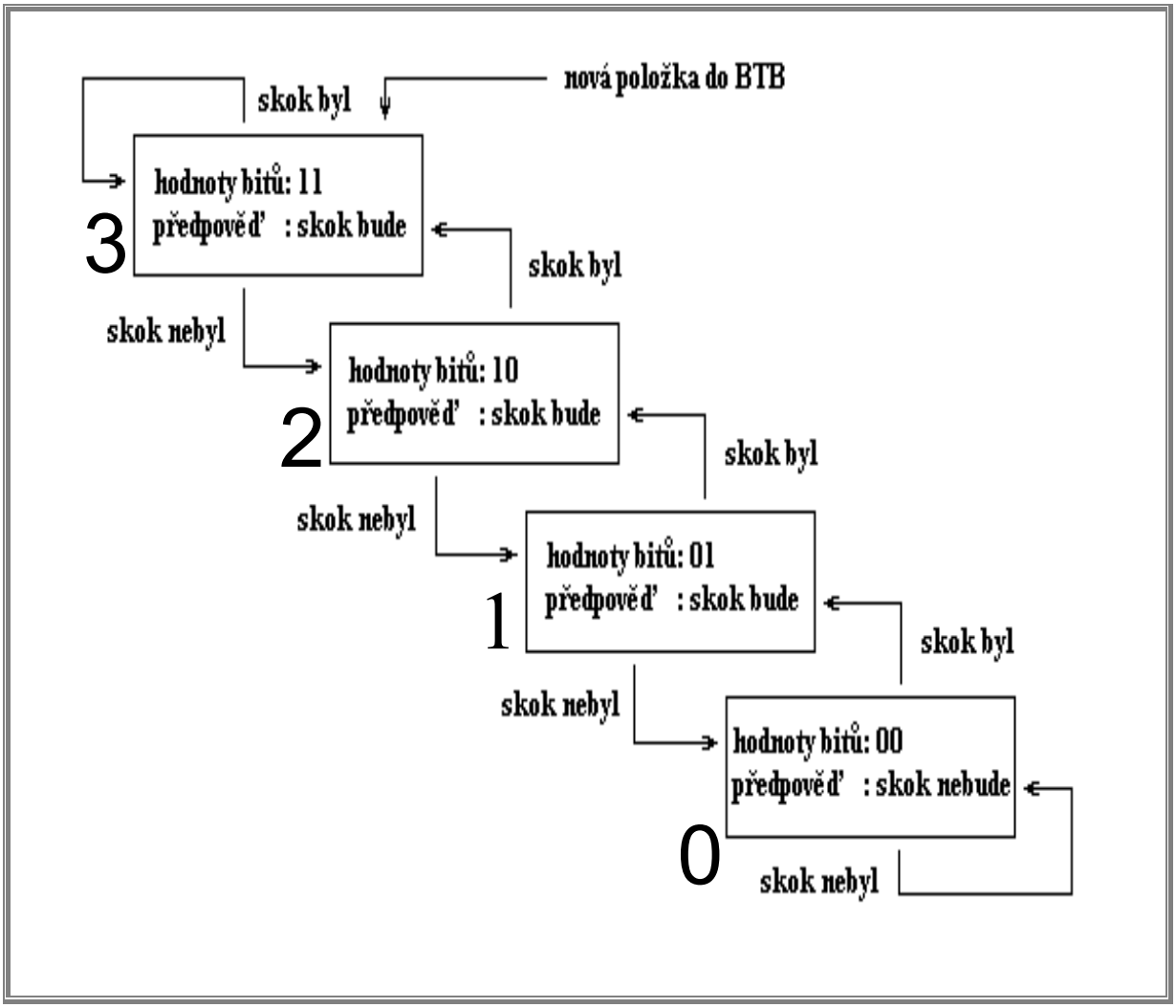
- Provádí zaokrouhlení výsledku.

Jednotka ER

- Analyzuje možné chyby.

Předvídání pokračování běhu programu

- Může vzniknout problém v případě, kdy má být provedena instrukce skoku => je nutné pokračovat ve zpracování na místě, kam ukazuje skoková instrukce - změna posloupnosti provádění instrukcí.
- Řešení: již první procesory Intel Pentium měly zabudováno tzv. **dynamické předvídání skoků** (Dynamic Branch Prediction) - technika pro odhadnutí, zda při dalším průchodu skoková instrukce skok způsobí nebo ne.
- Dynamické předvídání se řídí tím, jak se program při předcházejících průchodech konkrétním bodem choval.
- **BTB** (Branch Target/Trace Buffer) - v ní jsou uchovány poslední instrukce, které způsobily skok, spolu s dvoubitovou informací, jež určuje dosavadní chování těchto instrukcí.
- O každé instrukci skoku je uložena informace o tom, jak se tato instrukce skoku „chovala“ (tzn. skok nastal/nenastal).
- Podle hodnot těchto bitů je také stanovena předpověď, zda instrukce skok způsobí či ne.



Obr. 7 Schéma funkce předvídání větvení využívané mikroprocesory Intel

- Princip funkce:
    1. Instrukce, která způsobila skok, je uložena do BTB spolu se dvěma bity, jejichž hodnoty jsou rovny 1 (stav 3).
    2. Tyto hodnoty při příštím průchodu programu přes tuto instrukci signalizují předpověď, že skok bude.

3. Pokud skok skutečně byl, hodnoty bitů zůstanou nezmodifikovány.
4. Pokud byla předpověď mylná a skok nebyl, jsou bity nastaveny na hodnotu 10 (stav 2), která opět signalizuje, že skok bude.
5. Podle toho, zda skok skutečně následuje nebo ne, jsou pak příslušným způsobem bity modifikovány (viz obrázek) a jejich hodnota signalizuje předpověď skoku.
6. Pokud při prvním výskytu instrukce skok nenastane, nastaví se hodnoty bitů na 00 (stav 0).
7. Při dalším výskytu této instrukce se předpokládá, že skok nebude – pokud se to splní, zůstane nastaven současný stav 0.
8. Pokud skok nastane, hodnota bitů se změní na 01 (stav 1) a předpověď se změní na „skok bude“.

- Je to vlastně samoučicí se mechanismus.