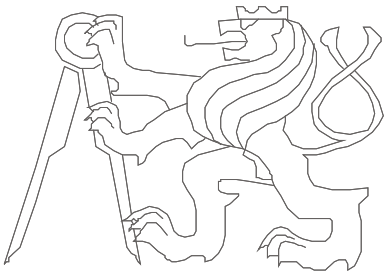
Pokročilé architektury počítačů

Spekulace, atd.



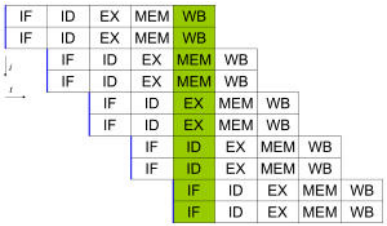
České vysoké učení technické, Fakulta elektrotechnická

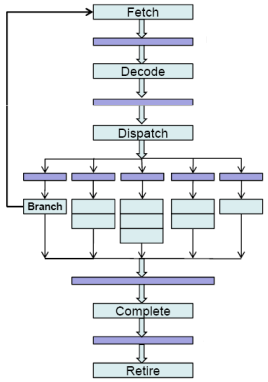
Osnova přednášky

- Spekulace, spekulativní provádění instrukcí,
- Přesné přerušení a cesty k jeho dosažení,
- Procesory VLIW,
  - Případová studie FFT, DCT
- Procesory EPIC,
- Využití datového paralelismu,
- Vektorové instrukce v ISA.

Připomenutí

- **Jak dosáhnout zrychlení sekvenčního výpočtu? Úpravami počítače:**
  - Superskalární architektura,
  - Predikce skoku (statická nebo dynamická).
- **Úpravami ALU:**
  - Zřetězením, ILP.
- **Další možnosti?**
  - Spekulativní provádění instrukcí,
  - Vektorovým zpracováním,
  - Paralelizací.





Spekulativní provádění instrukcí

- Spekulace je v této souvislosti
  - Spuštění kódu, jehož výsledek se „může hodit“ (může se použít).
- Potíž – skutečnost může dopadnout jinak.
- Spekulativní provádění instrukcí je formou optimalizace výkonu.

- Může ale mít (negativní) vliv na spotřebu procesoru.

U spekulace ale pozor

- Musí být splněna korektnost provádění programu
- To znamená: program splnil
  - Podmínku 1 - dosahuje správný výsledek (podle sekvenční sémantiky), tedy jako při provádění procesorem bez jakéhokoliv zřetězení,
  - Podmínku 2 - generuje stejná přerušení (výjimky) jako procesor bez jakéhokoliv zřetězení.
- Pro splnění korektnosti existují následující postačující podmínky:

Postačující podmínky

- Podm. A – respektování datových závislostí
  - instrukce čekají na jejich vyřešení
- Podm. B – respektování řídicích závislostí
  - skok se neprovede, dokud není známa adresa příští instrukce
- Podm. C – Přerušení je přerušením přesným.
- Při spekulativním provádění se částečně nebo dočasně narušují Podm. A a B.
- Korektnost provádění programu ale splněno být musí, Podm. C trvá.

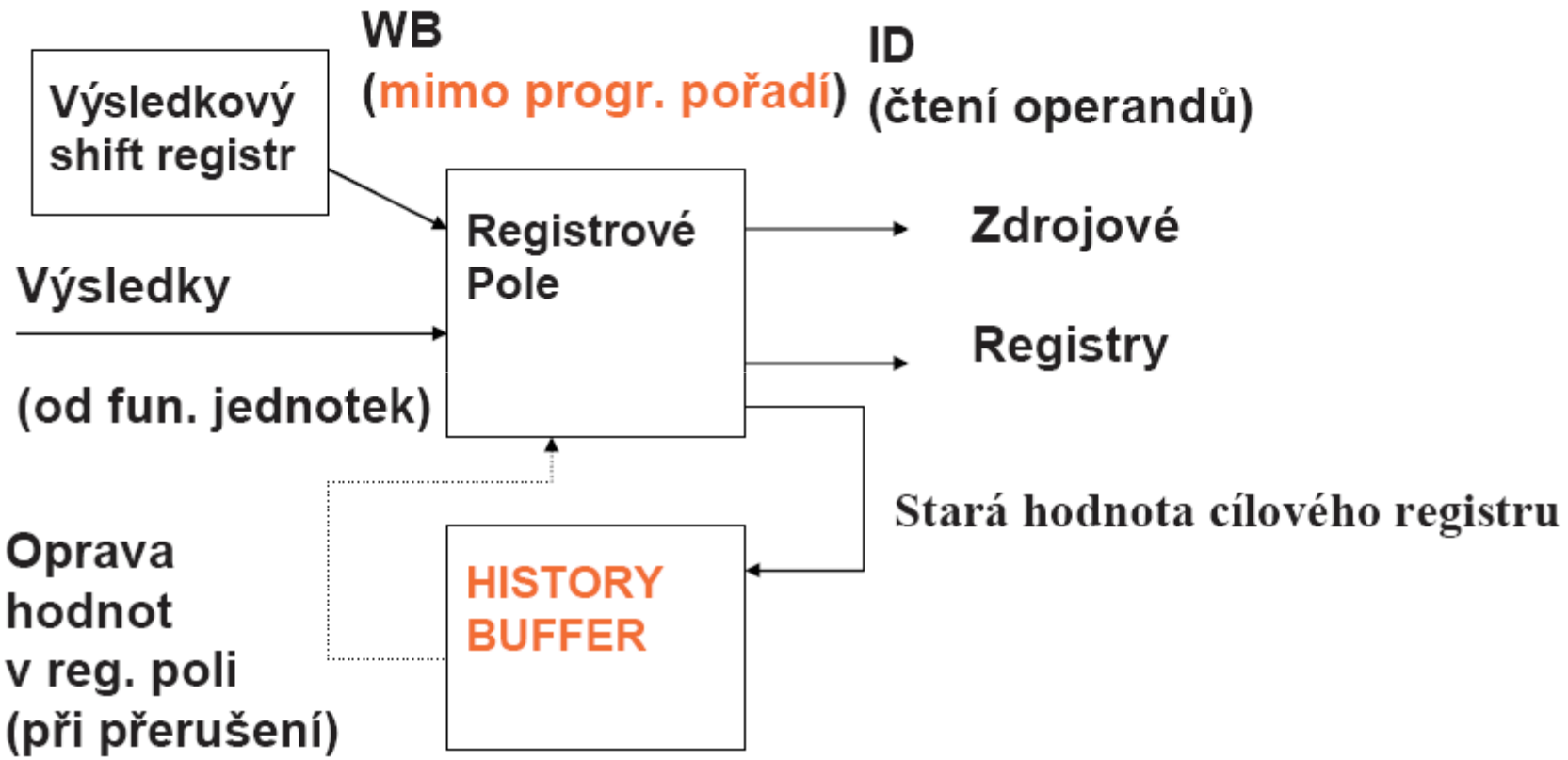
Využití spekulace

- Moderní procesory s paralelismem na úrovni instrukcí (ILP) užívají
- řídicí spekulace
  - u podmíněných skoků (dnešní standard) a
- datové spekulace (Load/Store spekulace)
  - Load se provede dříve, než je známa adresa předchozích Store (např. Itanium, Power 5, Core 2)

Chybná spekulace?

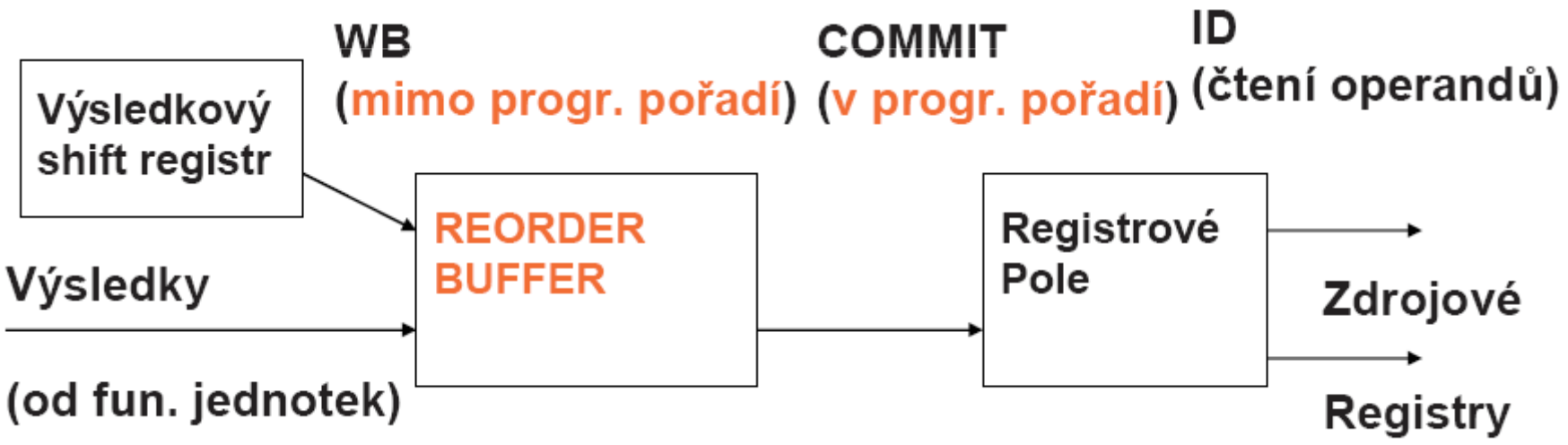
- Musí nastat
  - Zotavení,
  - Restart.
- Obě akce jsou relativně nákladné (až desítky ztracených taktů).
- Spekulaci a zotavení podporují dvě HW fronty
  - History Buffer HB - M88110
  - Reorder Buffer RB – ostatní procesory.

Řešení - History Buffer HB



Řešení - Reorder Buffer RB

- Výsledky instrukcí zapíšeme ve stupni WB do REORDER BUFFERU,
- z něj ve fázi COMMIT zapíšeme obsah do registrového pole až pokud
- všechny předchozí instrukce byly dokončeny.



COMMIT vysvětlíme později

History Buffer vs Reorder Buffer

- Výhody HB
  - Hodnoty z history bufferu se nemusejí forwardovat (není asociativní prohledávání HB) => jednodušší forwarding.
- Nevýhody HB
  - Nutnost rekonstruovat stav registrů sekvenčním procházením HB,
  - Dodatečný čtecí port pro registrové pole (problém se škálováním pro superskalární procesory).
- Výkonnost (za běhu) obou metod je srovnatelná, při přerušení však má HB značnou nevýhodu. Opět zde je problém s instrukcí STORE.

Přesné přerušení a cesty k jeho dosažení

Přesné přerušení

- Přerušení (opakování pojmu)
  - je metoda pro asynchronní obsluhu vnější události/í. Procesor přeruší sekvenční sémantiku vykonávání instrukcí, přejde na obsluhu a pak se vrátí a pokračuje v činnosti předchozí.
- Výjimka (vysvětlení pojmu)
  - je přerušením (obsluhou neplánované události) vyvolaným událostí uvnitř (v procesoru). Jiné označení pro vnitřní přerušení. Patří sem ale také nedefinovaná instrukce, systémové volání, apod.
- Přesné (Precise Exception)
  - Přerušení v sekvenčním stroji je vždy přesné.

Příklady přerušení a výjimek

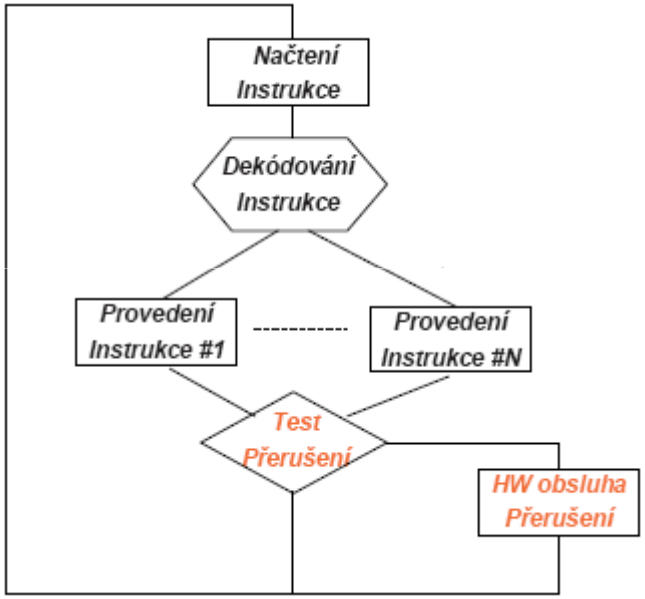
- Žádost V/V zařízení o obsluhu (I/O Device Request),
- Volání služby OS (Supervisor Call SVC),
- Trasování programu (Breakpoint),
- Přerušení AJ (aritmetické přeplnění, nenaplnění, dělení nulou,…),
- Výpadek stránky (Page Fault),
- Porušení ochrany paměti (Page Fault),
- Nezarovnaný přístup k paměti (Misaligned Memory Access)
- Nedefinovaná instrukce (Undefined Opcode),
- Chyba HW (parita, ECC, pokles napájení,…)

Klasifikace typů přerušení (jen pro ilustraci možných pohledů)

- Vnější vs. vnitřní, jindy se říká
- Přerušení vs. výjimka,
- Hardwarové vs. softwarové,
- Synchronní vs. asynchronní,
- Žádané uživatelem vs. vynucené,
- Maskovatelné vs. nemaskovatelné,
- V instrukci vs. mezi instrukcemi,
- Možnost návratu vs. ukončení s chybou.

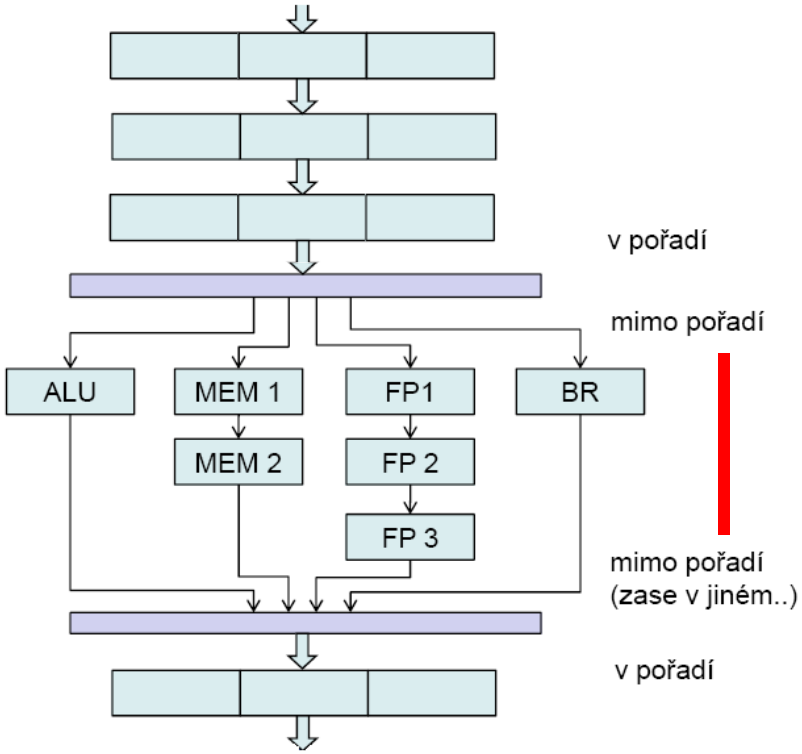
Běžný postup zpracování přerušení

- Sekvenční – historický
  - Procesor zpracovává instrukce v určeném pořadí
  - Vykonávání instrukcí se nepřekrývá
  - Vykonávání programu se může přerušit jen při přechodu mezi instrukcemi



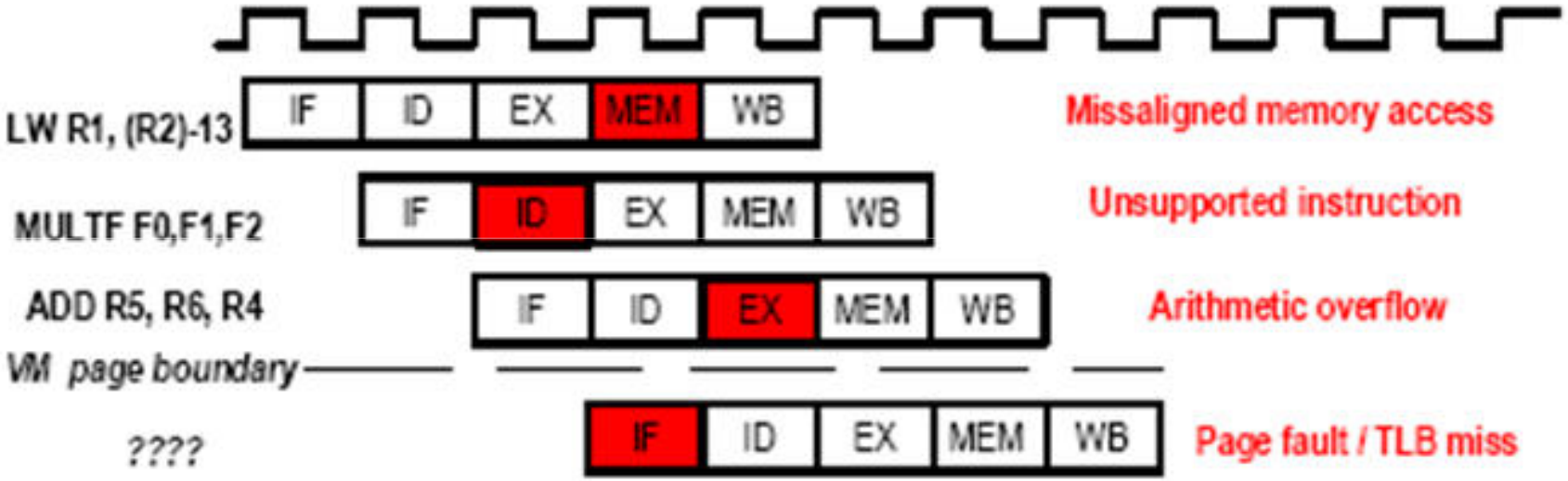
Specifika přerušení u proudově pracujícího procesoru

- K přerušení může dojít v každém taktu,
- Současně může dojít k přerušení z více zdrojů,
- Pořadí výskytu přerušení nemusí odpovídat pořadí instrukcí v programu,
- Stav procesoru nemusí být okamžiku přerušení konzistentní se sekvenční sémantikou vykonávání.



Příklad programu se vznikem více přerušení

Příčina



V některých taktech tu nastala 2 různá přerušení (přesněji) výjimky. Mohly nastat až dokonce 4! Řešení?

Je konkrétní přerušení přesné? Ano, pokud

- 1. procesor obslouží přerušení v programovém pořadí,
  - To nemusí odpovídat pořadí jejich objevení se v proudu (pipeline).
- 2. stav procesoru při jejich obsluze je sekvenčně konzistentní.
  - To znamená: instrukce před zdrojem přerušení musí být dokončeny a instrukce za zdrojem přerušení nesmí stav procesoru a paměti změnit.

Příklad

- Příčiny výjimek v našem celočíselném DLX
- **IF**
  - Výpadek stránky, nezarovnaný přístup, chyba ochrany paměti.
- **ID**
  - Neznámá instrukce.
- **EX**
  - Aritmetické přerušení z některé příčiny.
- **MEM**
  - Výpadek stránky, nezarovnaný přístup, chyba ochrany paměti.
- **WB**
  - Přerušení nemůže nastat.

Přirozená metoda řešení problému – Existence Commit

- Stupeň potvrzení (Commit) v proudovém zpracování.

- Stupeň potvrzení je takový stupeň proudu, že všechna přerušení mohou vzniknout nejpozději v tomto stupni.
- Po stupni potvrzení už víme, že instrukce nebude přerušena.

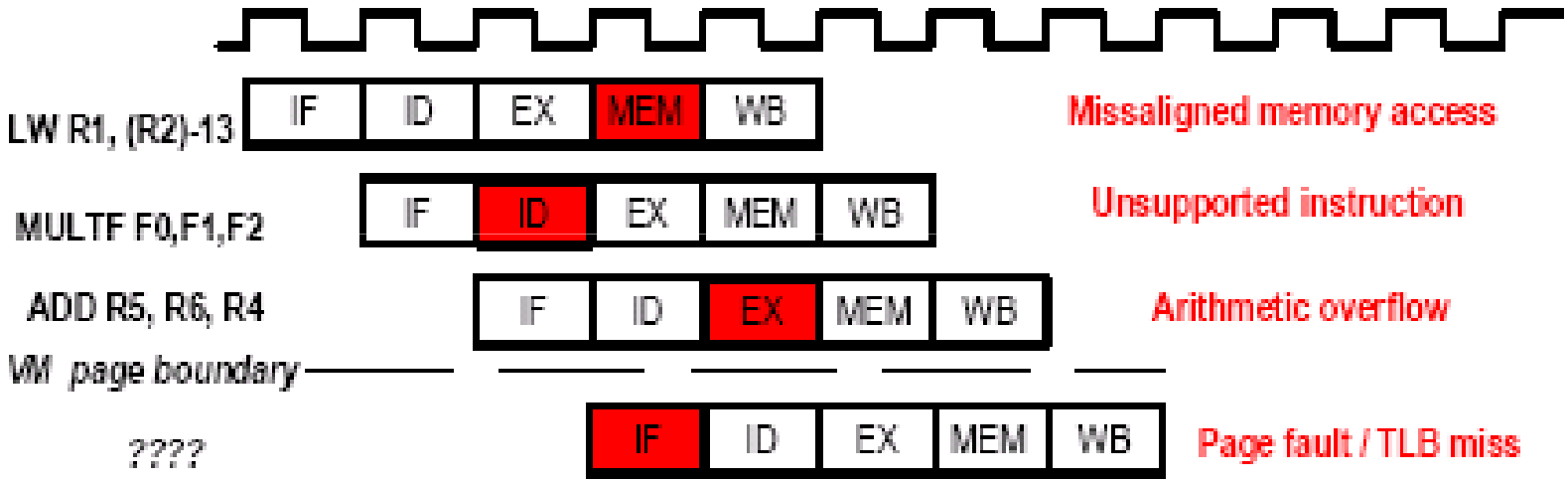
Podmínky pro přesné přerušení jsou splněny:

- Všechny instrukce dorazí do stupně potvrzení ve správném sekvenčním pořadí.
- Instrukce nemění stav procesoru ani paměti před stupněm potvrzení.

Princip implementace přesného přerušovacího systému

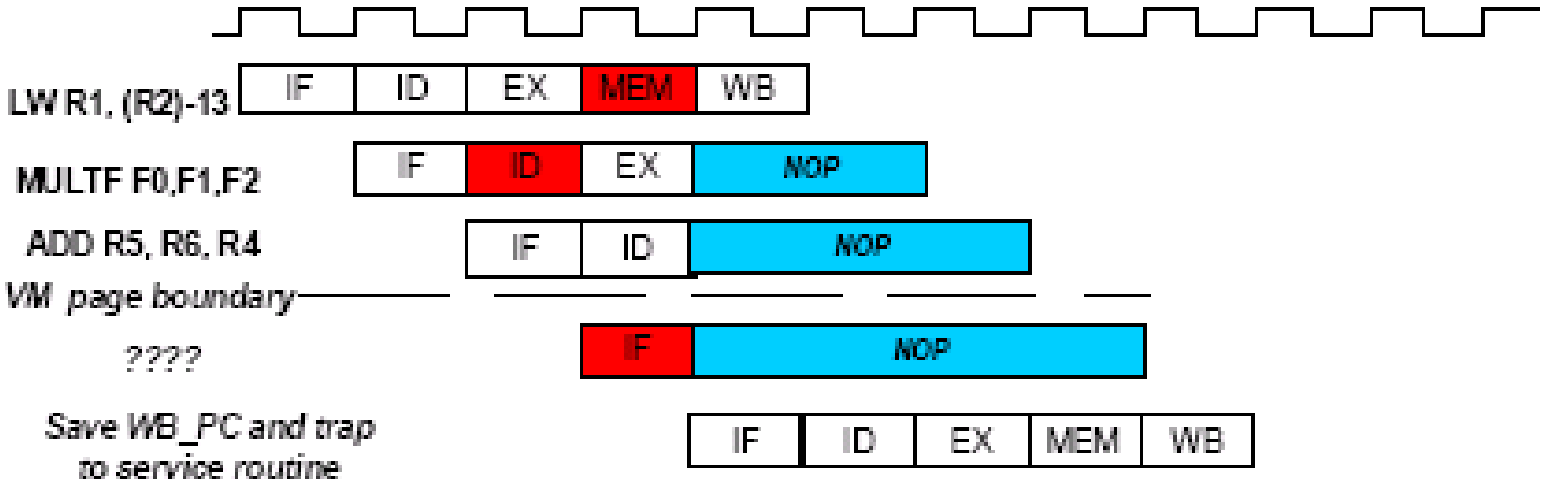
- Požadavky se akumulují ve speciálním HW a žádost společně s jejím PC postupuje v proudu až do stupně potvrzení.
- Procesor žádost testuje u každé instrukce až ve stupni potvrzení, přerušení v rámci instrukce se obsluhují v pořadí jejich vzniku.
- Obsluha začíná zrušením všech instrukcí v proudu před stupněm potvrzení a dokončí se instrukce za stupněm potvrzení.

Řešení u dříve uvedeného příkladu



Bude-li ale stupněm potvrzení (commit) u tohoto procesoru stupeň MEM ...

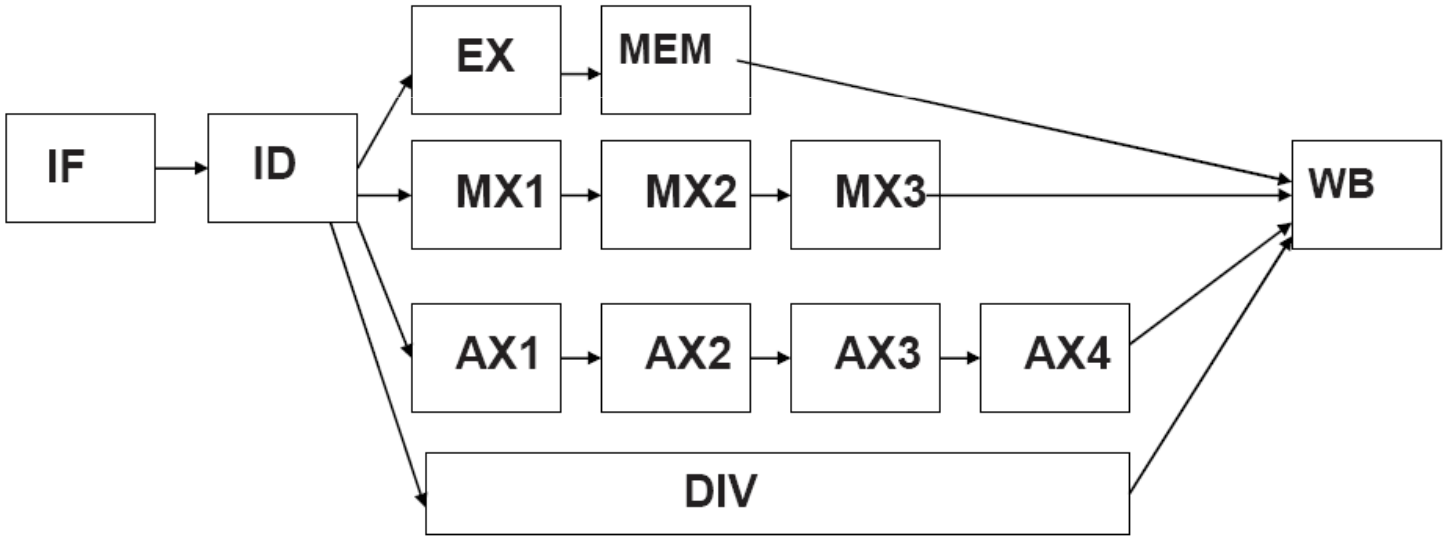
Pokračování příkladu



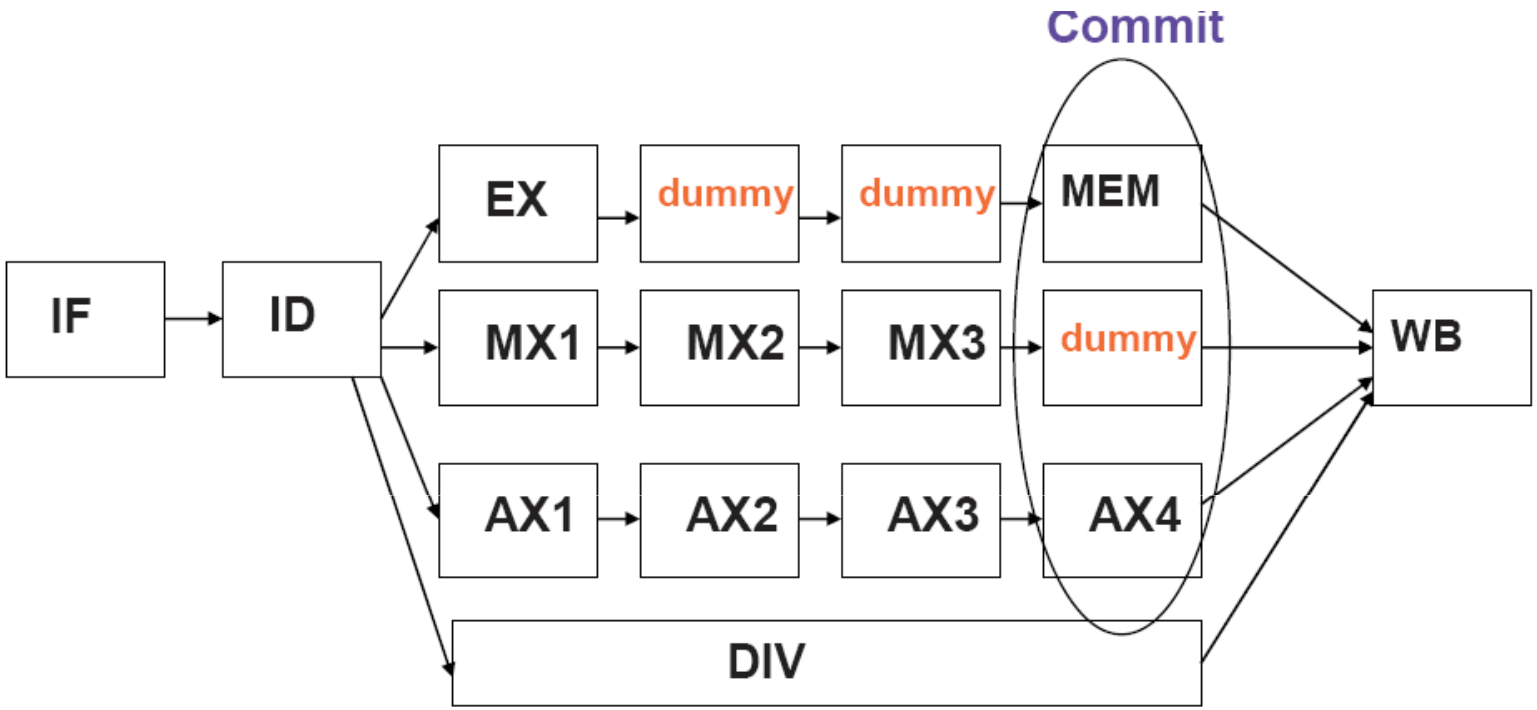
- Tento procesor má schopnost přesného přerušení, neboť
  - Do MEM dorazí v programovém pořadí a
  - Před stavem MEM instrukce nemění stav procesoru ani paměti.

Problém vyřešen?

- Ne, jsou složitější případy.
- Třeba ten, kdy nemají proudy stejnou délku.



Metoda linearizace proudu



Má řadu nevýhod:

- Zpomalení rychlých proudů, zbytečný HW,
- Nutnost speciálního ošetření dlouhých instrukcí (zde DIV),)
- Aj.

Jsou i jiné metody řešení

- Pštrosí (ignorování problému),

- Spouštění instrukcí s dokončováním v programovém pořadí (výsledkový Shift Registr)
- Reorder Buffer (viz dříve v této přednášce),
- History Buffer (dtto),
- Future File.

- Pozn. Zajímavé, ale podrobný výklad je nad rámec tohoto předmětu.

Procesory VLIW

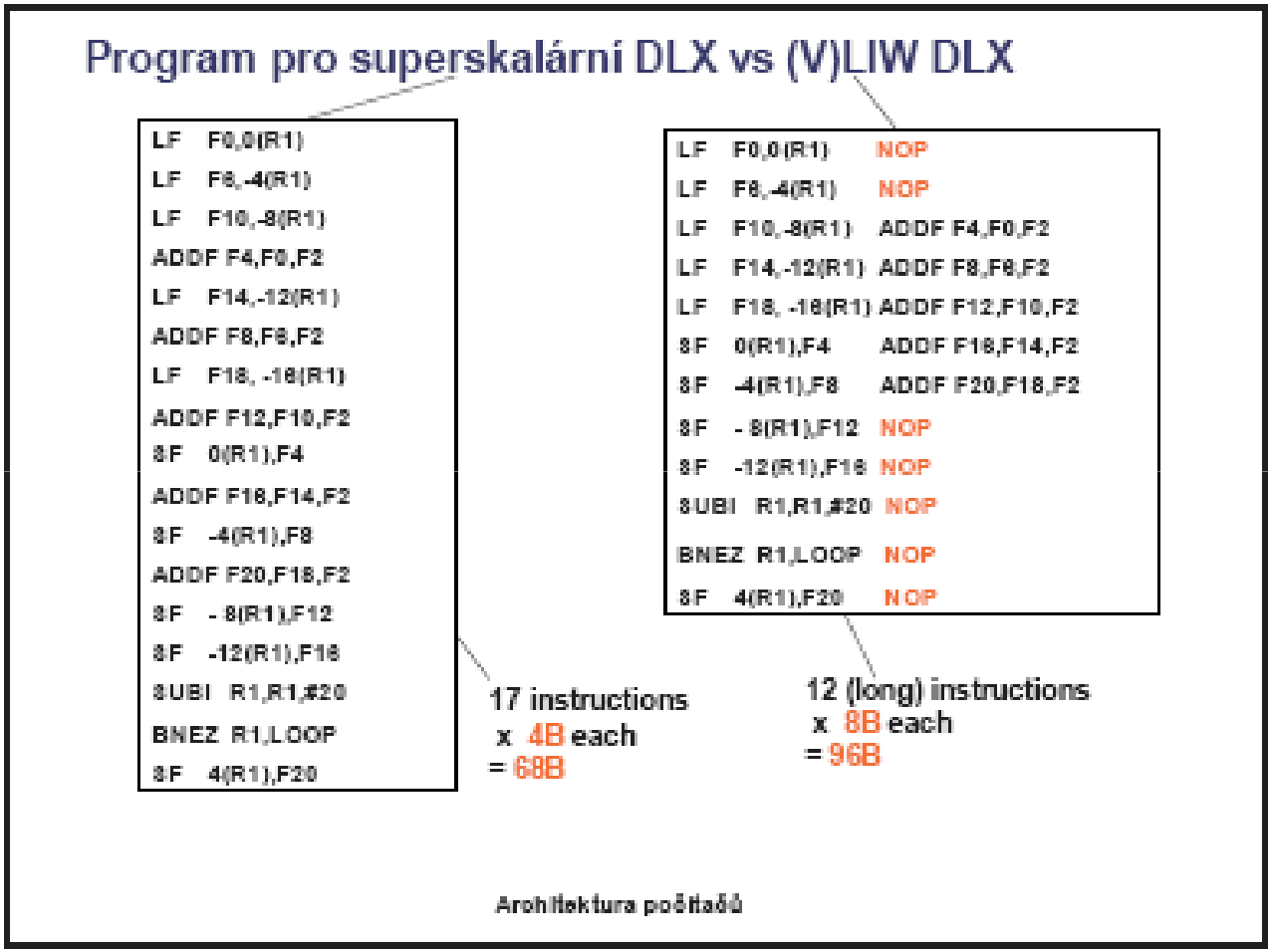
Procesory VLIW

- Very Long Instruction Word (typicky několik instrukcí)
- VLIW architektura umožňuje paralelní zpracování (původních několika instrukcí) jednou novou instrukce.
- Paralelně zpracovatelné instrukce jsou naplánovány předem.
- Při kompilaci.
- VLIW je vlastně příkladem třídy MIMD.

VLIW se zřetelem na přesné přerušení

- Sémantickou jednotkou pro akceptování přerušení zůstává instrukce (velmi dlouhá, či spíše široká)
- Pevný formát instrukce obsahuje kód několika operací, které se ale mohou provést paralelně.

Příklad

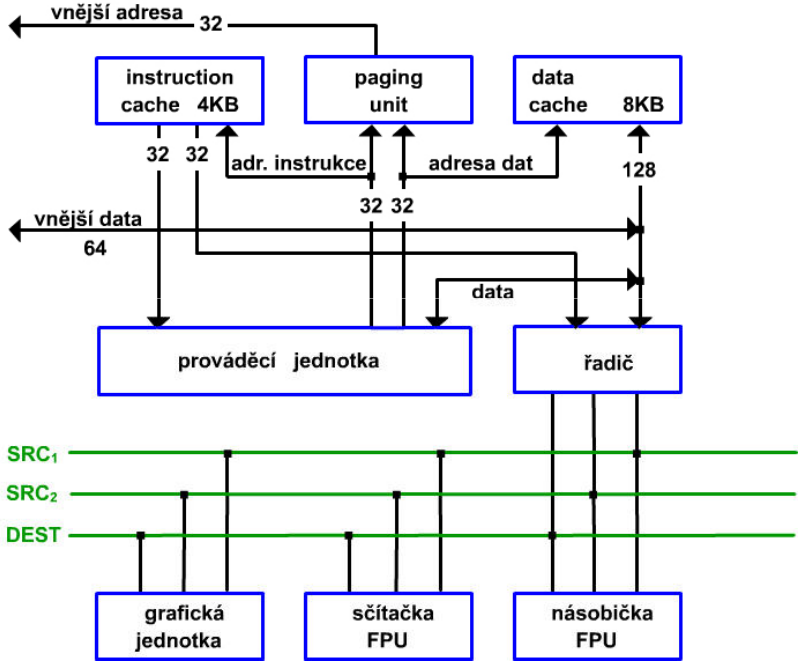

Program pro superskalární DLX vs (V)LIW DLX
LF F0,0(R1)
LF F8,-4(R1)
LF F10,-8(R1)
ADDF F4,F0,F2
LF F14,-12(R1)
ADDF F8,F8,F2
LF F18, -16(R1)
ADDF F12,F10,F2
SF 0(R1),F4
ADDF F16,F14,F2
SF -4(R1),F8
ADDF F20,F18,F2
SF - 8(R1),F12
SF -12(R1),F16
SUBI R1,R1,#20
BNEZ R1,LOOP
SF 4(R1),F20
17 instructions
x 4B each
= 68B
LF F0,0(R1) NOP
LF F8,-4(R1) NOP
LF F10,-8(R1) ADDF F4,F0,F2
LF F14,-12(R1) ADDF F8,F8,F2
LF F18, -16(R1) ADDF F12,F10,F2
SF 0(R1),F4 ADDF F16,F14,F2
SF -4(R1),F8 ADDF F20,F18,F2
SF - 8(R1),F12 NOP
SF -12(R1),F16 NOP
SUBI R1,R1,#20 NOP
BNEZ R1,LOOP NOP
SF 4(R1),F20 NOP
12 (long) instructions
x 8B each
= 96B
Architektura počítačů

Příklad VLIWu

i860 XP
A80860XP-50
L4190197
SX657
(M)(C)1988 1990

94075906AC
MALAY
A 419

Core Frequency:	50 MHz
Board Frequency:	50 MHz
Data bus (ext.):	64 Bit
Address bus:	64 Bit
Transistors:	2,500,000
Circuit Size:	0.80 µ
Introduced:	1988
Manufactured:	week 19/1994
L1 Cache:	16+16 KB
CPU Code:	N11
Intel S-Spec:	SX657
Package Type:	Ceramic PGA-262



Moderní DSP

- Superskalární zpracování
- Frekvence více než 1 Ghz
- Dvouúrovňové cache s až 8 MB
- SIMD
- VLIW – až 8 instrukcí za instrukční cyklus
- Speciální jednotky pro výpočet celé FFT

Případová studie FFT, DCT

Diskrétní Fourierova transformace

- Namísto spojité funkce s periodou T pracujeme se vstupním vektorem s N prvky.

$$X(\omega) = \int_{-\infty}^{\infty} x(t).e^{-j\omega t}\,dt, \quad j = \sqrt{-1}, \quad \omega = 2\pi f,$$

$$\Downarrow$$

$$X(k) = \sum_{i=0}^{N-1} x(i).e^{\frac{-j2\pi ik}{N}}, \quad k = 0,1,2,\ldots,N-1$$

$$x(i) = \frac{1}{N}\sum_{k=0}^{N-1} X(k).e^{\frac{-2\pi ik}{N}}, \quad i = 0,1,\ldots,N-1$$

Diskrétní Fourierova transformace

Nahradíme-li $w^{ik} = e^{-j2\pi ik/N}$, můžeme zapsat maticově $[X_k] = [w^{ik}][x_i]$. Matice W pro N prvků bude mít tvar:

$$W_N = \begin{bmatrix} w_N^0 & w_N^0 & w_N^0 & \cdots & w_N^0 \\ w_N^0 & w_N^1 & w_N^2 & \cdots & w_N^{(N-1)} \\ w_N^0 & w_N^2 & w_N^4 & \cdots & w_N^{2(N-1)} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ w_N^0 & w_N^{(N-1)} & w_N^{2(N-1)} & \cdots & w_N^{(N-1)(N-1)} \end{bmatrix}$$

Díky ortogonalitě platí $[x] = [W^T][X]$.

DFT - Diskrétní Fourierova transformace

Jelikož funkce e^z je $j2\pi$ periodická, bude platit že $w^{ik} = w^{ik \bmod N}$. Řada koeficientů bude mít proto stejnou hodnotu. I přesto je časová složitost řádu N^2. Př. pro N=8:

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^2 & W^4 & W^6 & W^8 & W^{10} & W^{12} & W^{14} \\ W^0 & W^3 & W^6 & W^9 & W^{12} & W^{15} & W^{18} & W^{21} \\ W^0 & W^4 & W^8 & W^{12} & W^{16} & W^{20} & W^{24} & W^{28} \\ W^0 & W^5 & W^{10} & W^{15} & W^{20} & W^{25} & W^{30} & W^{35} \\ W^0 & W^6 & W^{12} & W^{18} & W^{24} & W^{30} & W^{36} & W^{42} \\ W^0 & W^7 & W^{14} & W^{21} & W^{28} & W^{35} & W^{42} & W^{49} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

$$W^0 = W^8 = W^{16} = W^{24}, \quad W^1 = W^9 = W^{25} = W^{49}, \quad W^2 = W^{10} = W^{18} = W^{42}, \quad W^3 = W^{35},$$
$$W^4 = W^{12} = W^{20} = W^{28} = W^{36}, \quad W^5 = W^{21}, \quad W^6 = W^{14} = W^{30}, \quad W^7 = W^{15}.$$

Table 2. Numerical Values of W_8^i, where i = 0, 1, ..., 7

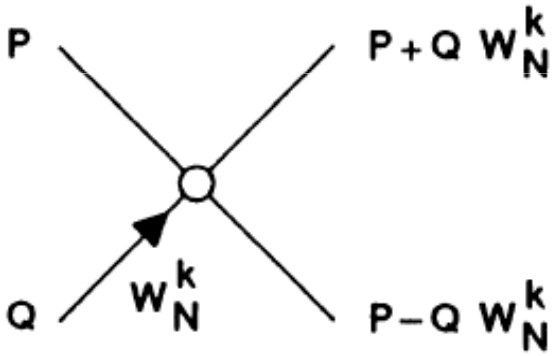
TWIDDLE FACTOR	VALUE
W_8^0	1
$W_8^1 = e^{-\pi/4}$	0.7071 − j 0.7071
$W_8^2 = e^{-\pi/2}$	− j
$W_8^3 = e^{-3\pi/4}$	− 0.7071 − j 0.7071
$W_8^4 = e^{-\pi} = -W_8^0$	− 1
$W_8^5 = e^{-5\pi/4} = -W_8^1$	− 0.7071 + j 0.7071
$W_8^6 = e^{-3\pi/2} = -W_8^2$	j
$W_8^7 = e^{-7\pi/4} = -W_8^3$	0.7071 + j 0.7071

FFT?

- Fast Fourier Transformation
  - je třídou efektivních algoritmů pro výpočet DFT.
- Popsali ji
  - J.W.Cooley a J.W.Turkey (1963) na 4 stránkách v odborném časopisu, nicméně to znamenalo převrat v DSP!
- Nejrozšířenější jsou
  - algoritmy pro $N=2^m$, m přirozené,
- možné jsou i jiné,
  - např. pro N prvočíselné.

Zvolíte-li $N=2^m$

- Přejde celý algoritmus na opakovanou aplikaci elementární transformace



Dvojkový motýlek

RADIX-2 BUTTERFLY

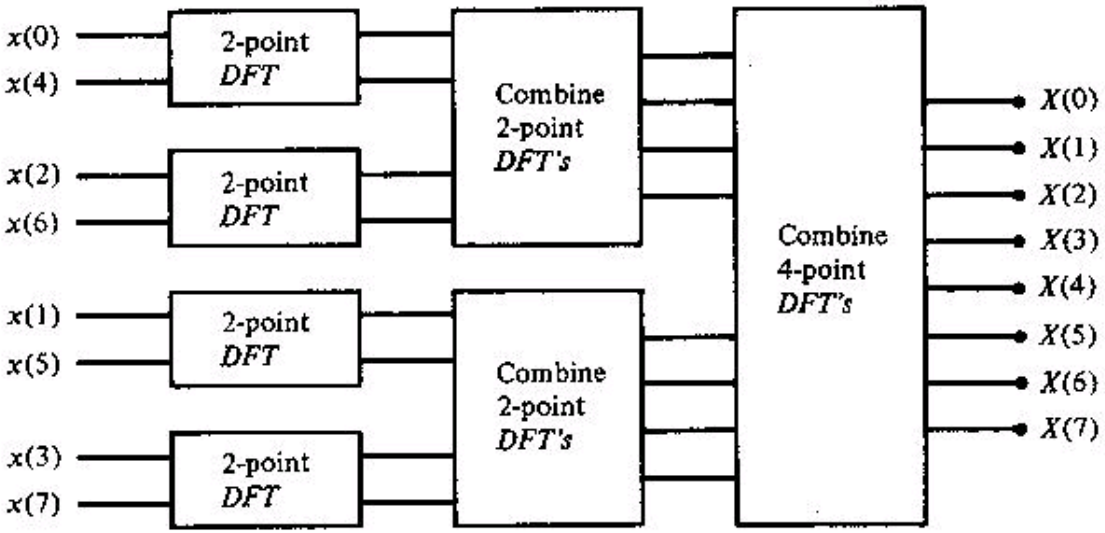
Jak? Aplikací postupu DIT nebo DIF! Co to znamená?

Základní myšlenka FFT

- Vychází se přirozeně z DFT.
- Čím delší je vstupní posloupnost, tím vyšší (kvadraticky) má DFT výpočetní náročnost.
- Ale: fázor (otáčecí vektor) *W* ve vzorcích pro DFT má zajímavé vlastnosti
  - je periodický a
  - symetrický.
- Sledujte obrázek.

Butterfly – motýlek

- tvar dataflow diagramu



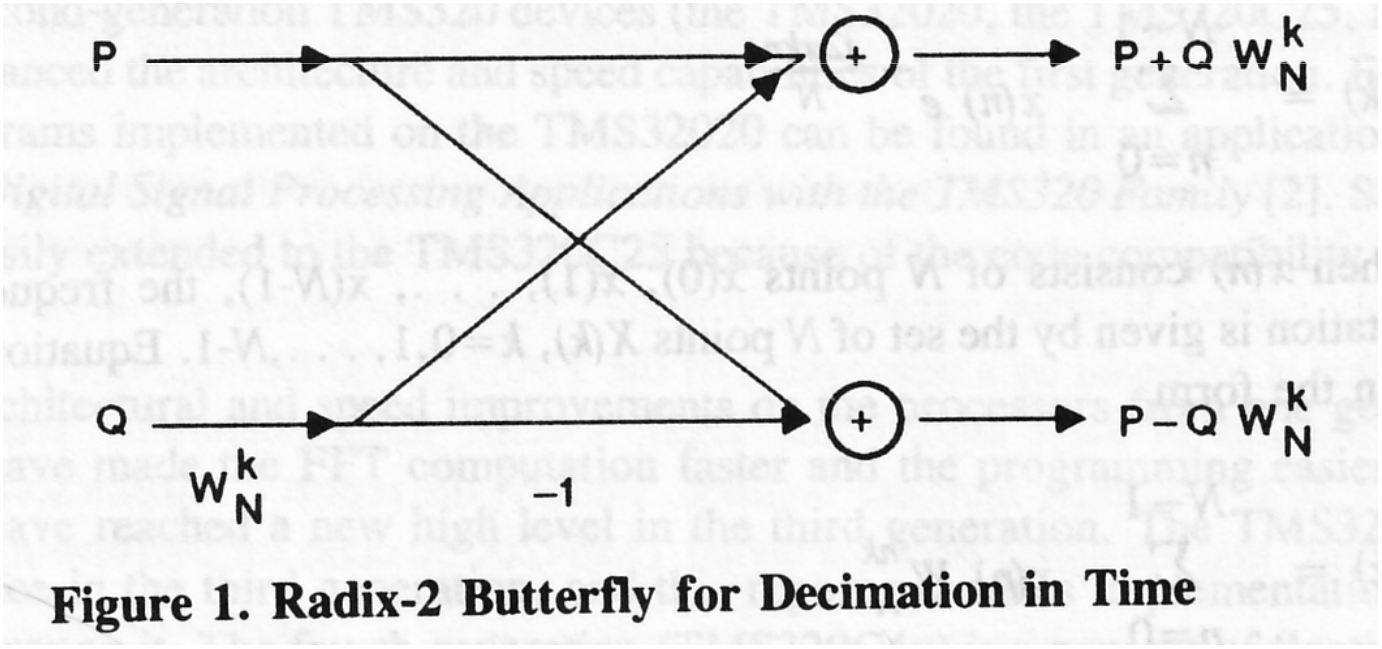
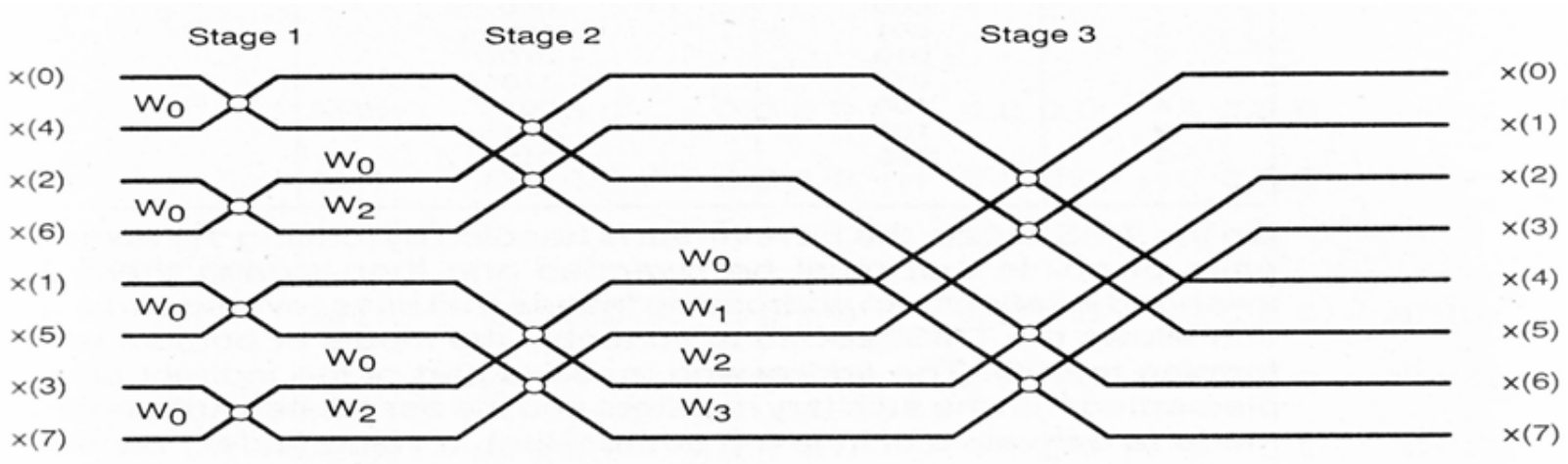


Figure 1. Radix-2 Butterfly for Decimation in Time

Technická podpora speciálních adresovacích režimů

Index	Bit Pattern	Bit-reversed Pattern	Bit-reversed
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7



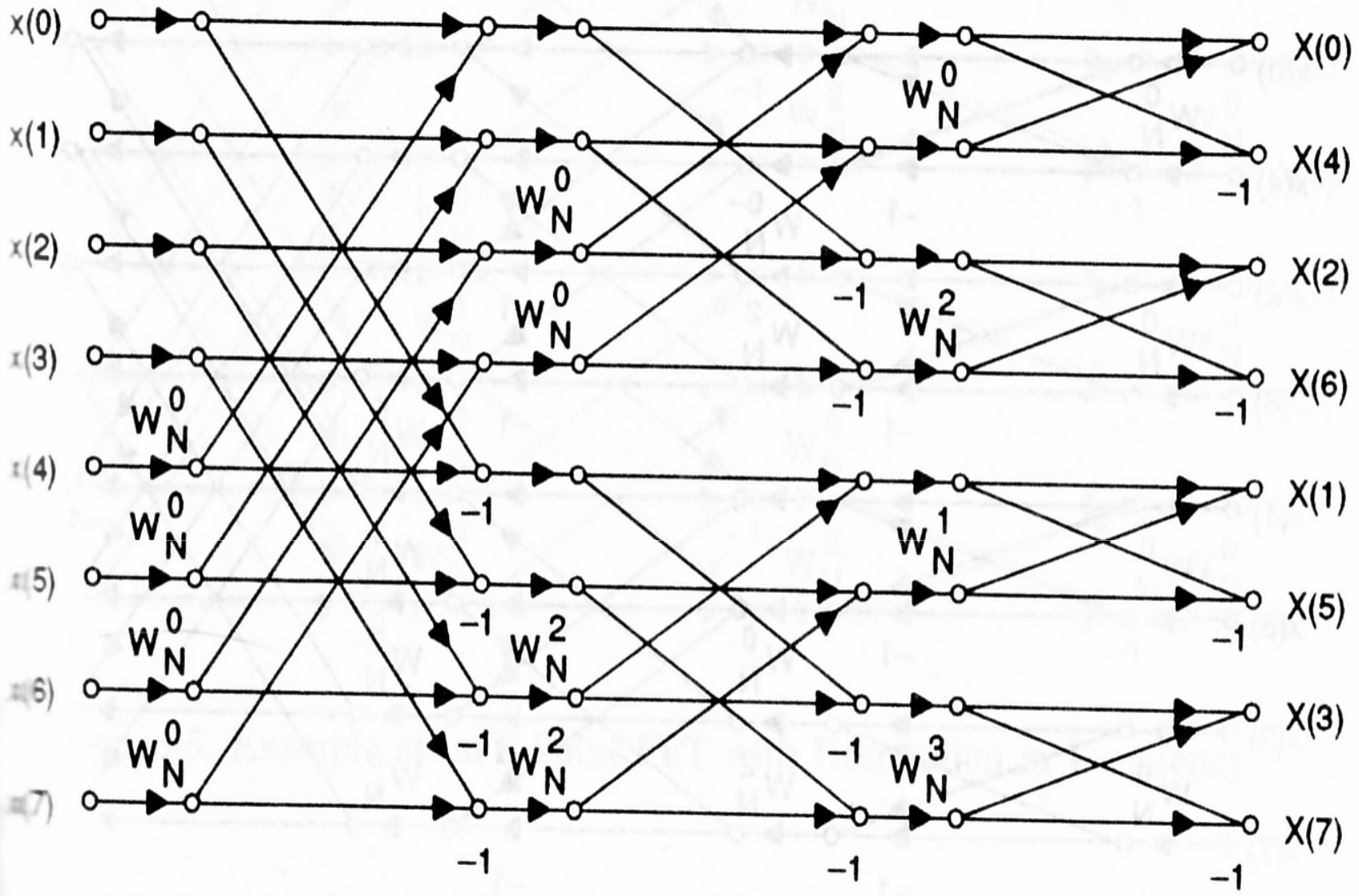


Figure 3. Example of 8-Point FFT with Decimation in Time.

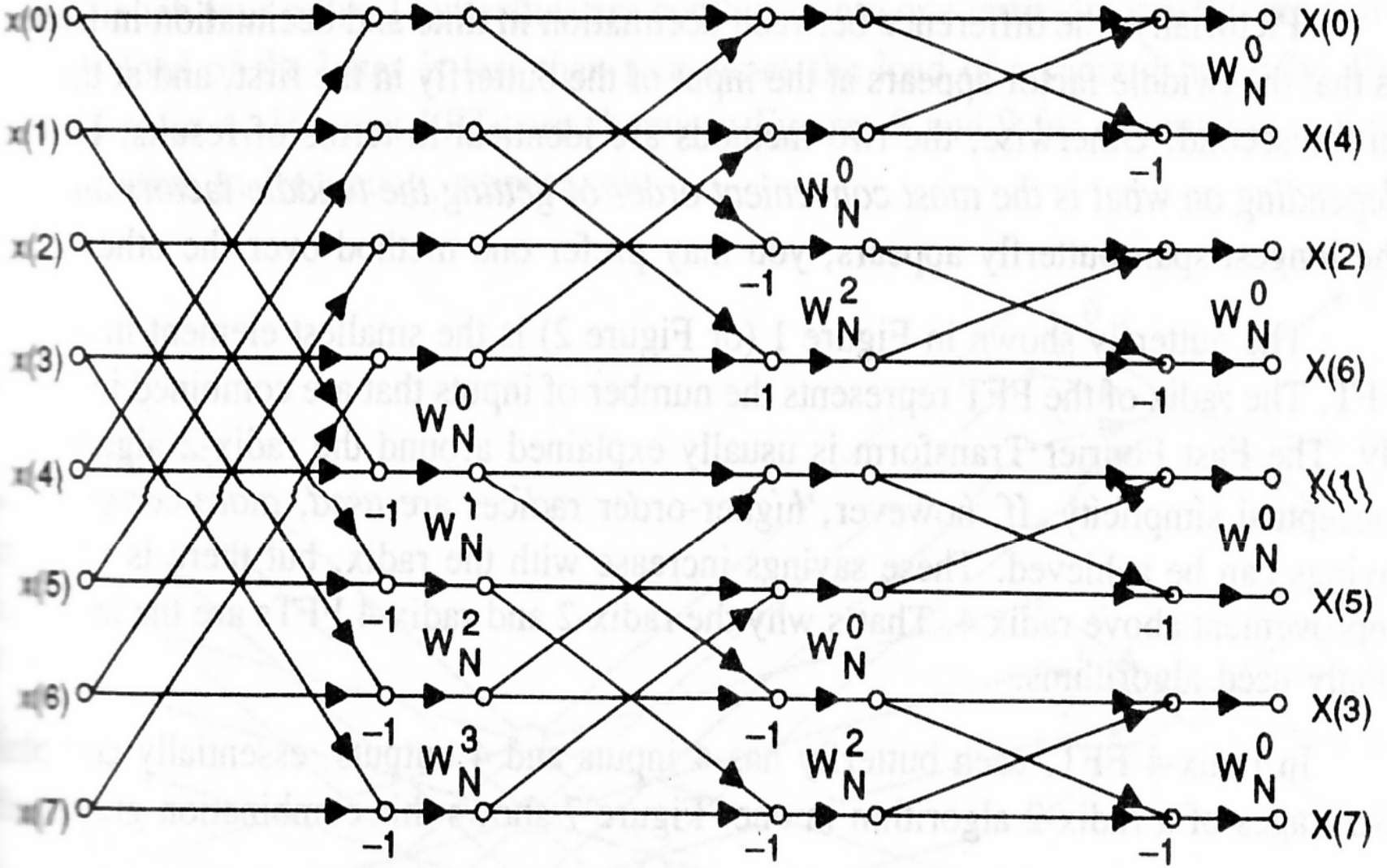


Figure 5. Example of an 8-Point FFT with Decimation in Frequency.

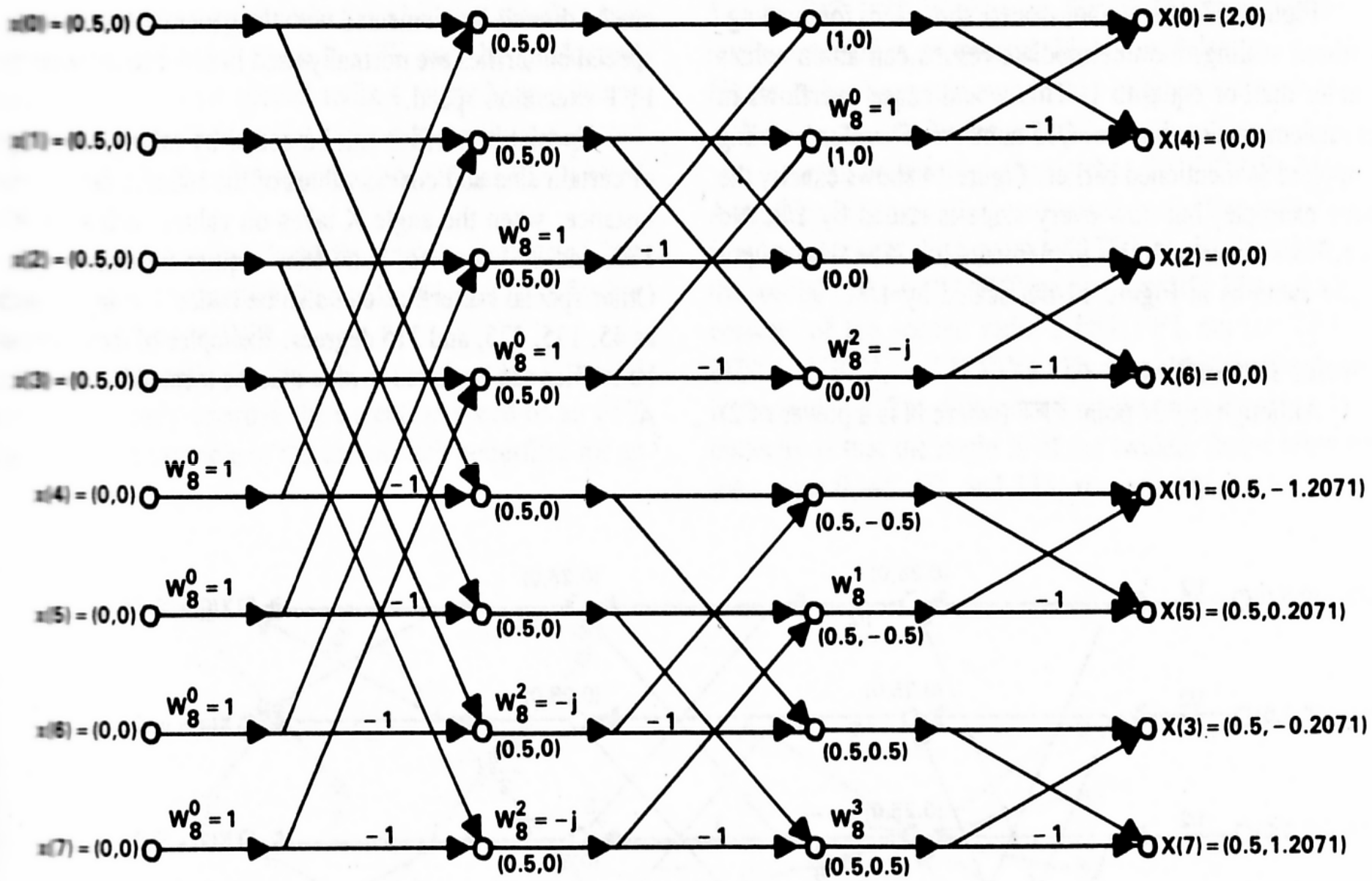


Figure 13. Numerical Example of an 8-Point DIT FFT without Scaling

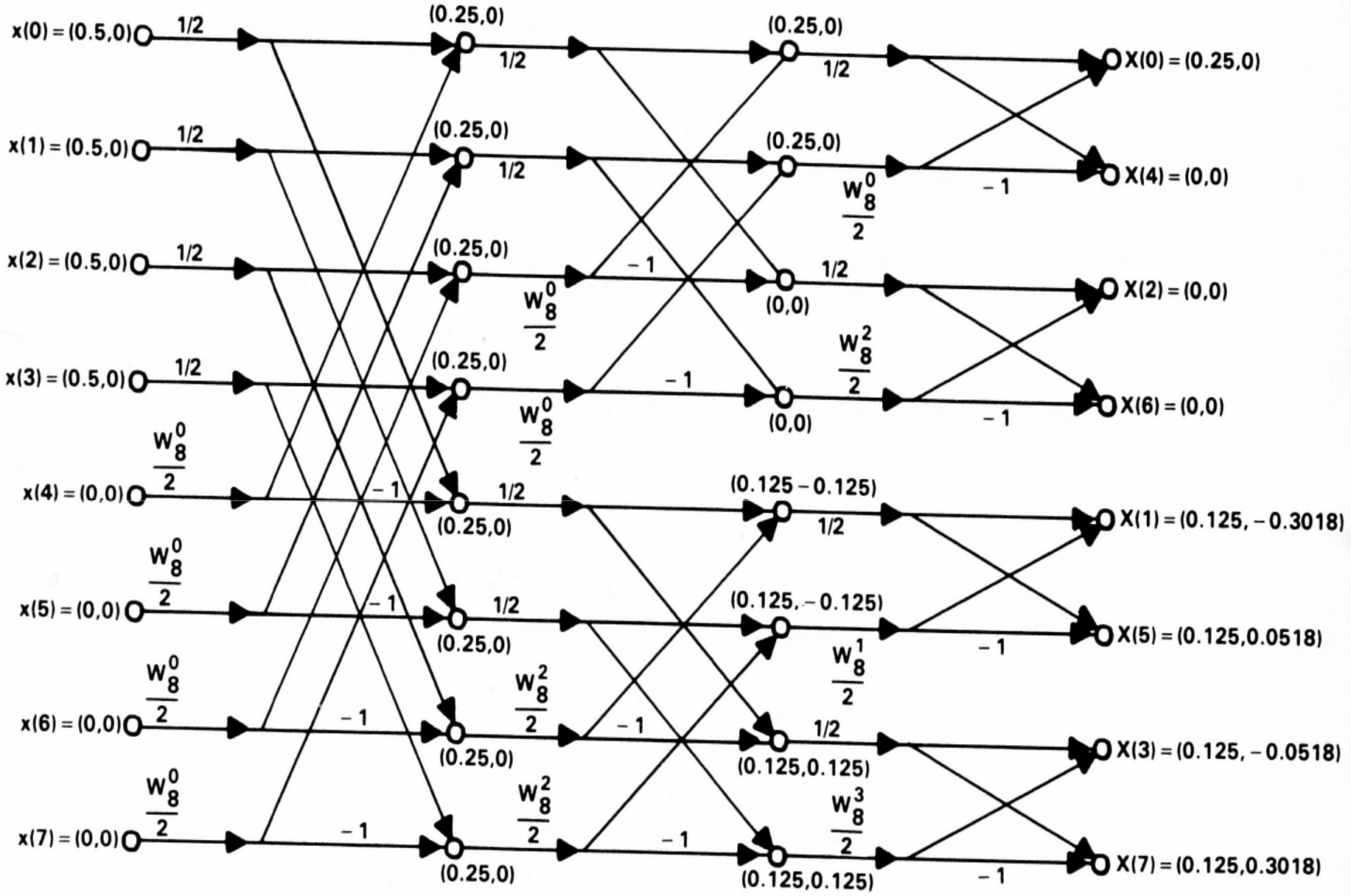


Figure 14. Numerical Example of an 8-Point DIT FFT with Scaling

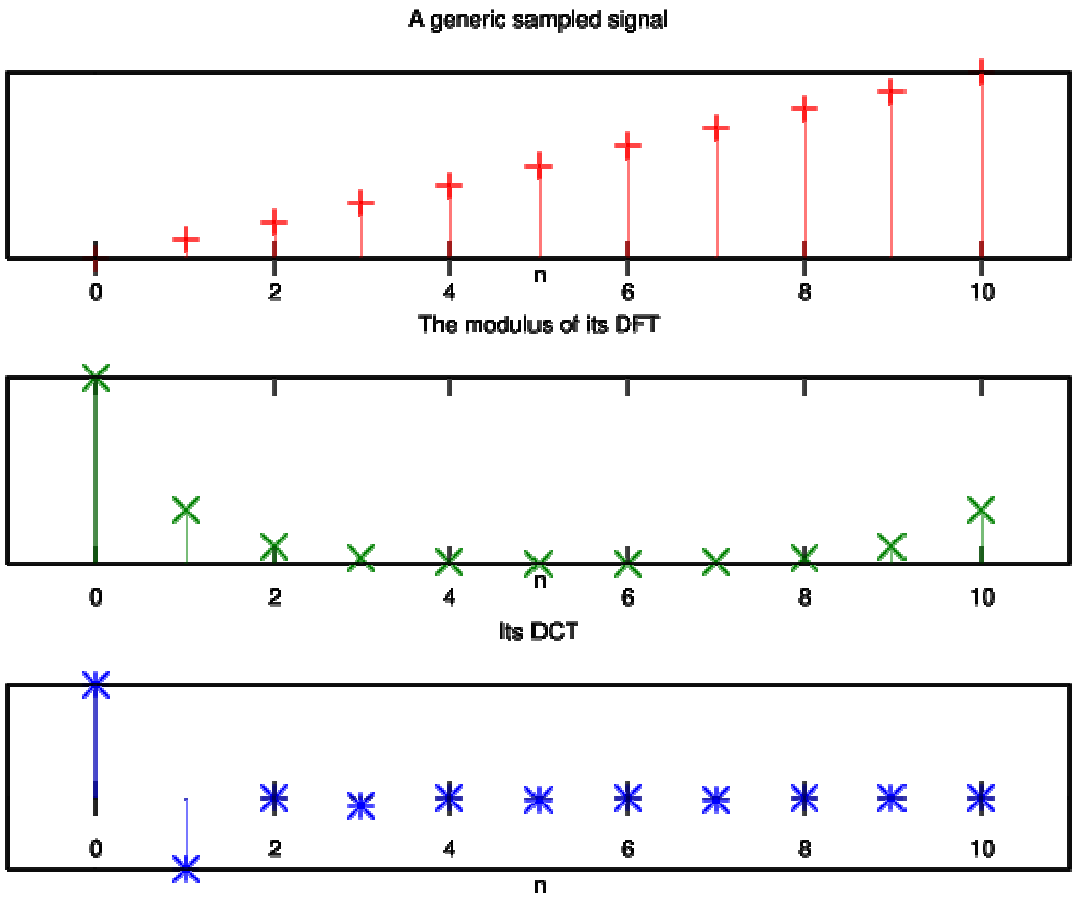
První srovnání DCT s DFT

- Proč transformovat signál už víme, ne?

- FT převod spojitého signálu z časové oblasti do frekvenční,
- DFT analogická transformace pro diskrétní průběhy.

- Nevýhoda DFT: pracuje s komplexními čísly.

Ilustrace souvislostí



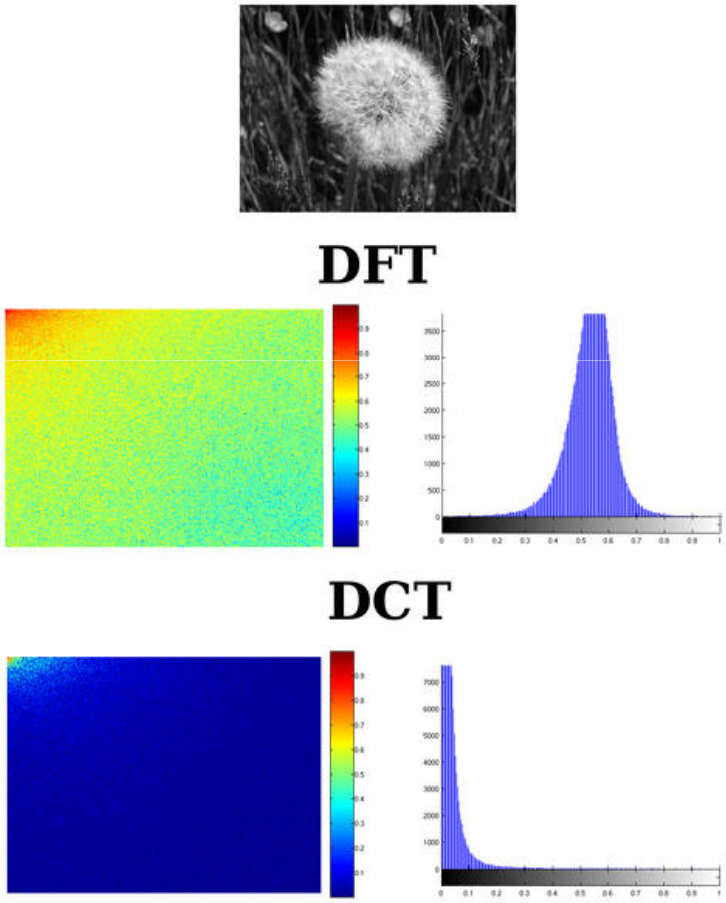
Původní signál,

jeho Fourierovo spektrum

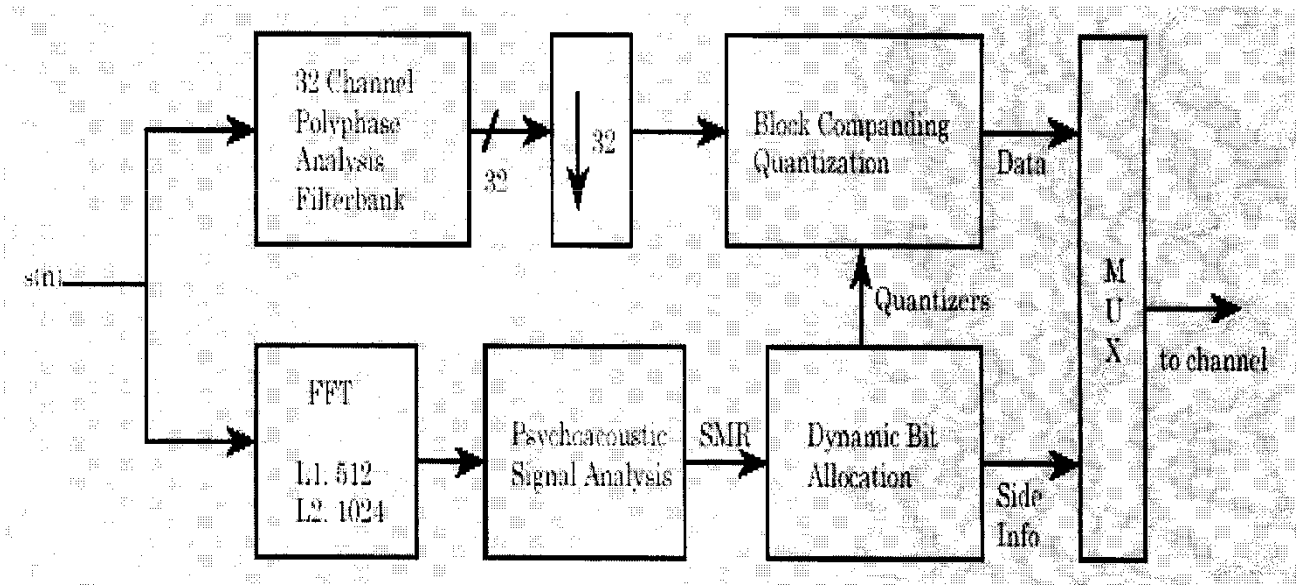
jeho DCT spektrum

Porovnání vlastností (pro 2D případ)

- Vlevo koeficienty spektra,
- vpravo histogram.
- Všimněte si: maximum informace je u DCT soustředěno v nejnižších frekvencích!



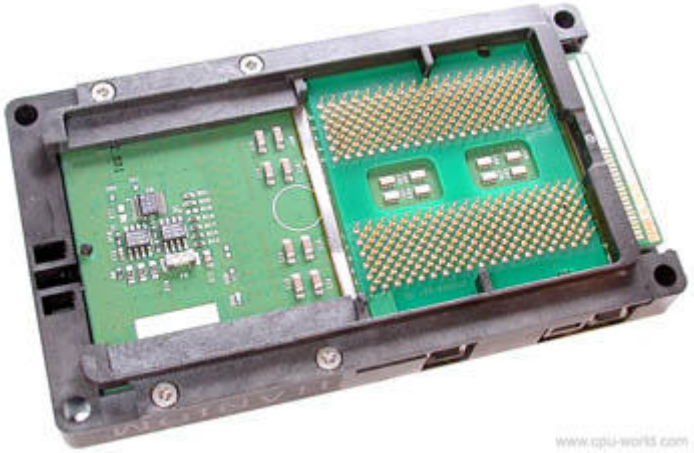
Aplikace FFT – například MP3/L1, L2

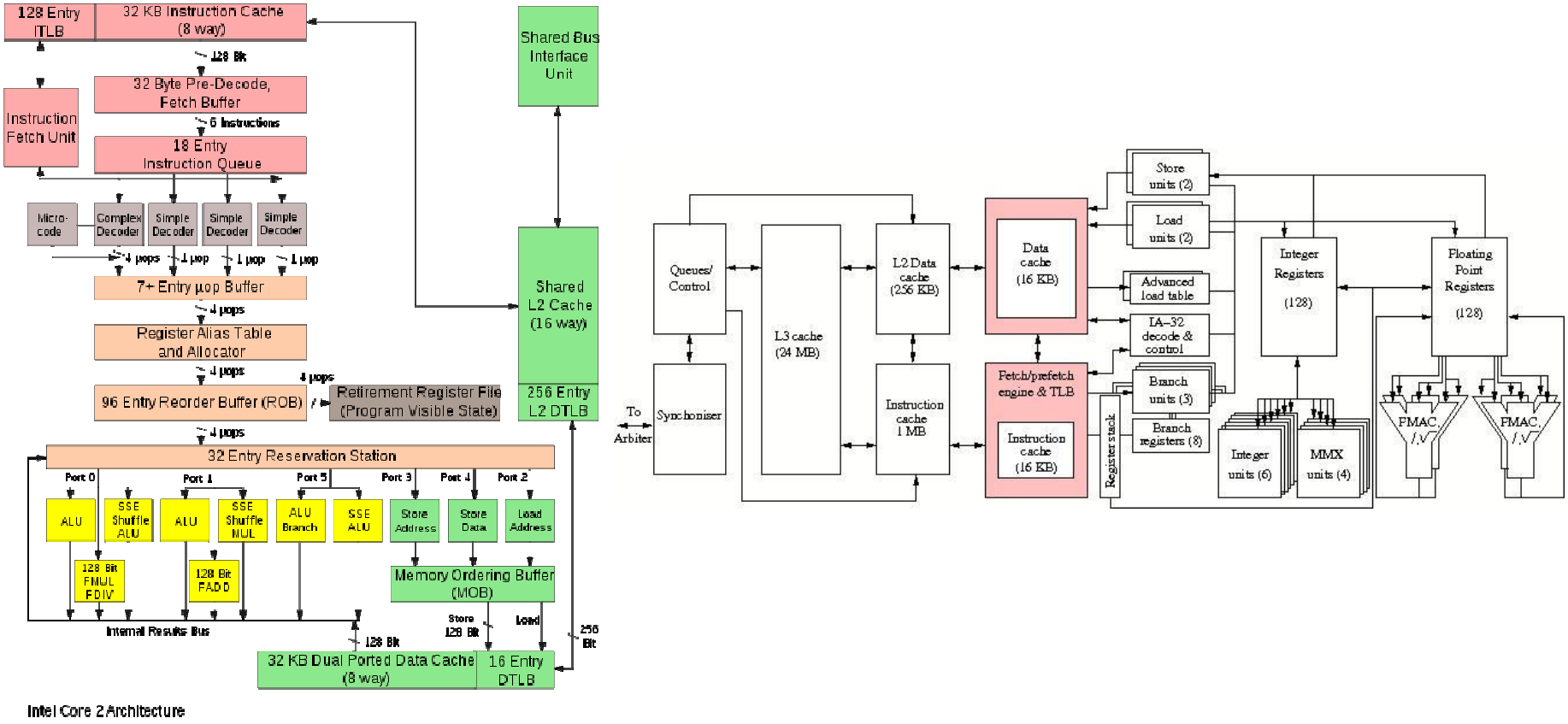


Procesory EPIC

Co je to EPIC

- Explicitly Parallel Instruction Computing
- Kořeny této architektury vyrůstají z VLIW
- Představitelem této architektury je **Itanium** (dřívější název **IA-64**).
- Jsou v ní implementovány dříve popsané metody jako spekulace, predikce skoků a přejmenování registrů.





Tento obrázek pochází z Wikimedia Commons

- Od architektury x86 (i x86-64) se dramatic-ky liší.
- Je založena na explicitní ILP, o paralelizaci se rozhoduje při překladu.
- Paralelně se provádí v jednom hodinovém taktu až 6 instrukcí.
- Nepotřebuje ale speciální HW pro zajištění odstranění hazardů.
- Další podrobnosti? Referát.

Využití datového paralelismu

Využití datového paralelismu, SIMD

- Single Instruction, Multiple Data, jedna ze tříd klasické Flynovy taxonomie.
- První široce užívanou implementací SIMD architektury bylo pro hry určené MMX (*MultiMedia eXtensions)* rozšíření pro x86.

Vektorové instrukce v ISA

Vektorové instrukce v ISA

- Do této skupiny patří i další rozšíření:
- 3DNow! Od AMD,
- SSE a další verze SSE2 a SSE3 od Intelu.