Principy překladačů

Architektury procesorů

Jakub Yaghob

Architektury procesorů

- Architektura procesoru představuje cílový jazyk
  - Platí pro překladače do kódu konkrétního procesoru
  - Ovlivňuje celý backend překladače, zejména generátor kódu
  - Může ovlivnit i tvar a instrukce mezikódu, nasazené druhy optimalizací nad mezikódem

Registry

- Nejrychlejší paměť
- Malý počet
  - x86 má 7 celočíselných 32-bitových
  - IA-64 má 128 celočíselných 64-bitových
- Různé druhy pro různé typy dat
  - Celočíselné, floatové, adresové, vektorové
- Různé druhy přístupu
  - Přímý
  - Zásobník (FPU)

= pomoc pro překladače: zvášť instr. pro čtení a zápis, ostatní instr. pracujou jen s registrama, ne s pamětí!!! (chovaj se tak všechny modern procesory krom IA32)

Instrukční sada

paměť je výrazně pomalejší, proto je dobře, když jsou tyhle instr. oddělený

- RISC reduced
  - Jednoduché instrukce, malý počet (např. bez dělení) ortogonální
- CISC complex
  - Složité instrukce, velký repertoár

hw instrukce, rychlejší než dělat to kódem ale složitější generátor kódu; nicméně knihovny se dělaj líp a jsou rychlejší

- Load-Execute-Store
- Ortogonalita mužu používat který registry chci
  - x86 má neortogonální instrukční sadu

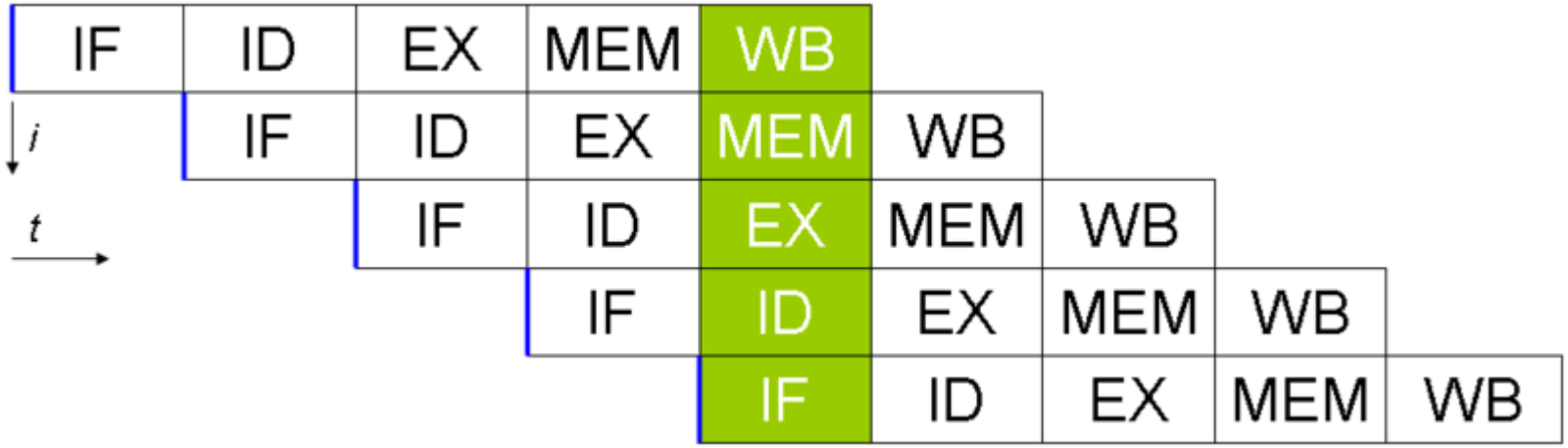
teď se to spojuje: jednoduchý instrukce se chovaj jako na RISC, tj. jsou zadrátovaný a prováděj se ďábelsky rychle :-)
složitý instrukce jsou tam taky (mikrokód)

IA32: transparentní pajplajna, programátor o ní vůbec neví
SPARC: zveřejněna existence a hloubka pajplajny, procesor sám NEOPRAVUJE špatné skoky, tj. n instrukcí po skoku se ještě vykonaj

Pipelining

ARM: hloubka 9

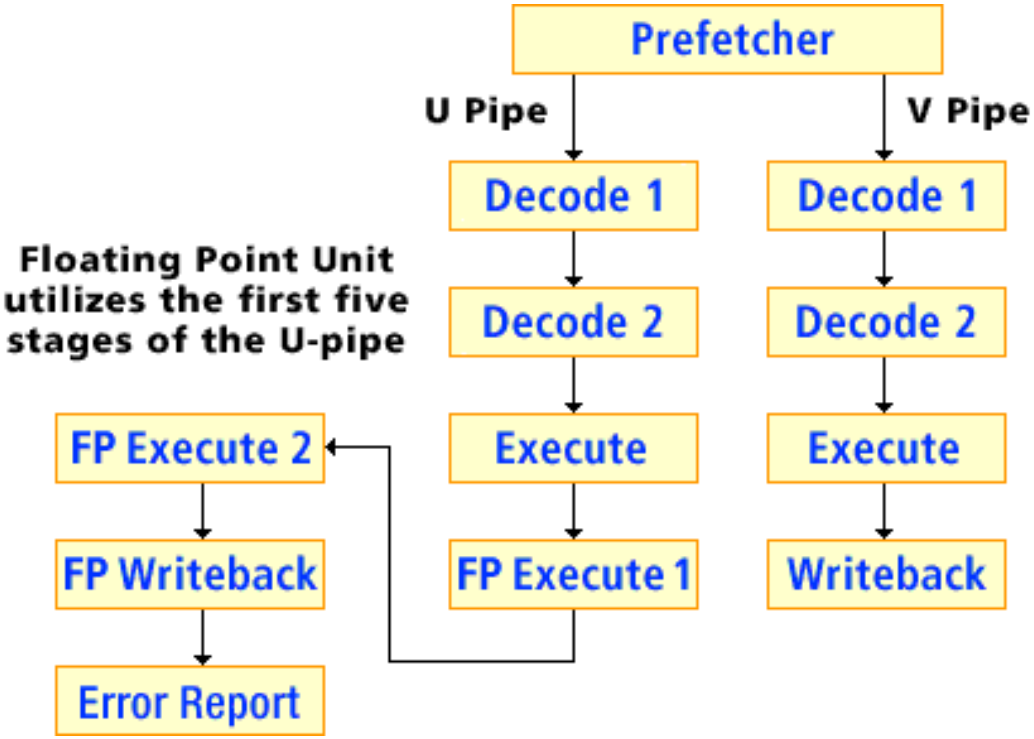
- Zahájení čtení a dekódování další instrukce před dokončením vykonávání předchozí instrukce
- Každá fáze a každá instrukce má svoje zpoždění (latency).
  - Problém při závislosti operandů (RAW) read after write

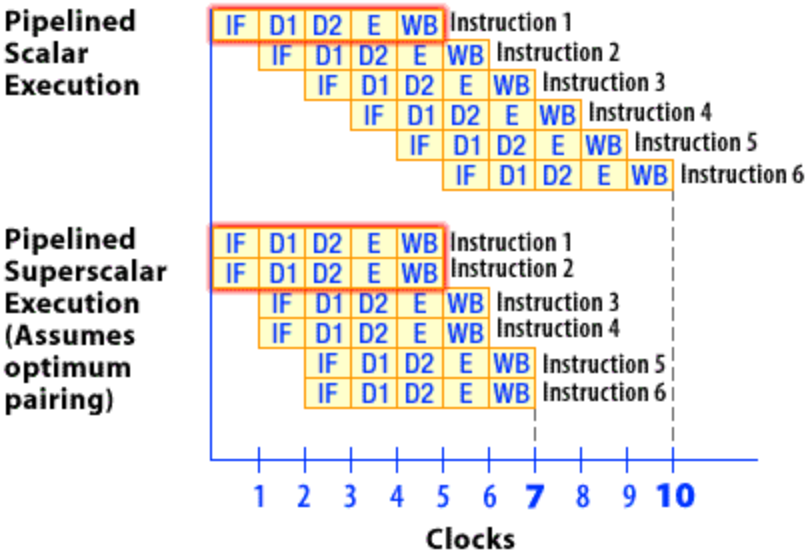


tady záleží na tom, v jakým pořadí tam ty instrukce posílám (některý nejsou párovat)

Superscalarita

- Více stejných jednotek schopných paralelního vykonávání instrukcí



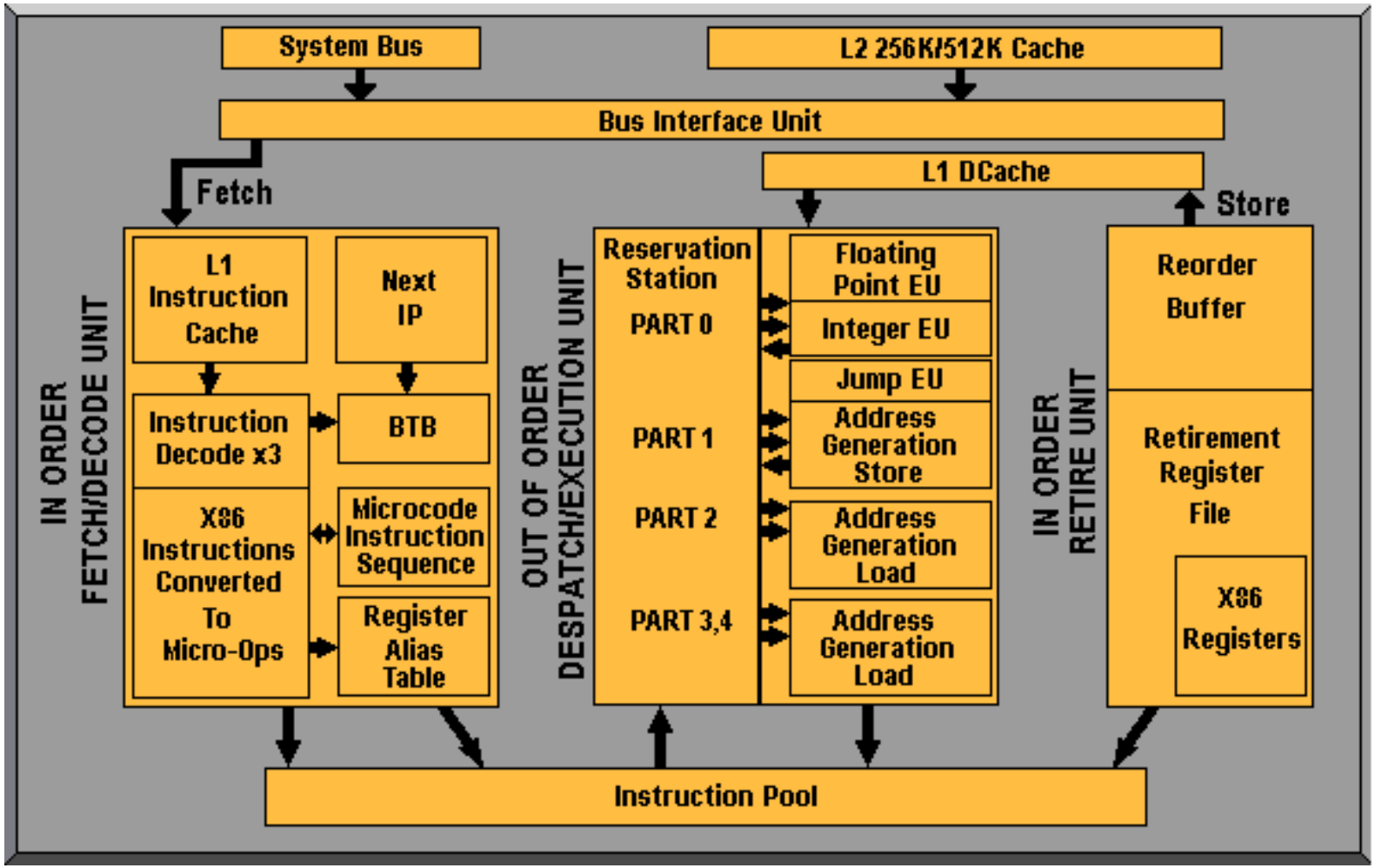


Out-of-Order Execution – 1 (vykonávání mimo pořadí)

- Načtení instrukce
- Zařazení do vyčkávací stanice
- Instrukce čeká na všechny svoje operandy
- Instrukce se vykoná ve své výkonné jednotce
- Výsledky se uchovají ve frontě
- Až se všechny starší instrukce zapíší do registrů, zapíše se výsledek této instrukce

Out-of-Order Execution – 2 (vykonávání mimo pořadí)

instrukce se načte, rozdrtí se na mikroinstrukce, obarví se barvou potřebné Execution Unit, a hodí se do bazénku; tam plave, dokud ji EU nevezme



Predikce skoků

- Hluboké pipeliny mají problém, pokud podmíněný skok není proveden
- Dynamická predikce skoků
  - BTB
  - CPU podle historie (vzory nějaké hloubky) odhaduje, jestli skok bude proveden nebo nebude
  - Pokud není žádná historie, použije se statická predikce
- Statická predikce skoků
  - Bez nápovědy
    - Skok vpřed se neprovede, skok vzad se provede
  - S nápovědou
    - Překladač odhaduje pravděpodobnost skoku
- Delay slot

Spekulativní vykonávání

musim udržet závislosti, to musim hlídat, detekovat, opravit…

- Vykonávání kódu, který nemusí být zapotřebí
- Významná disproporce mezi rychlostí CPU a paměti
- Typické využití je značné předsunutí čtecích operací
- CPU provádí i odsouvání zápisových operací
  pomocí různejch cache to odložit co nejvíc to jde
- Paměťové bariéry
- Kódová spekulace = explicitní, pro jednodušší procesory, spekulace provádí překladač
  - Kontrola, zda se provedla = instrukce CHECK, procesor IA64 má spešl obvody na detekci problémů (i třeba přes pointer aliasing)
- Datová spekulace
  - Předsunutí i v případě nevyjasněného pointer-aliasingu

SIMD instrukce

= jednou instrukcí provádím operaci nad celým vektorem dat (Intel: 256b)

- Někdy také multimediální instrukce
- Celočíselné i floatové datové typy
- Vektorizace výpočtu už v mezikódu nebo až při generování kódu
- Značné zrychlení výpočtu důležitý pro numerický výpočty (operace s maticema)
- Obtížná detekce možnosti vektorizovat

  opět nutná podpora překladače

VLIW very long instruction word

pevný formát → zrychlení dekódování

embedded procesory a IA64

- Kódování instrukcí
  - Šablony IA-64 128b, 3 instrukce o 41b + šablona 5b
- ILP (Instruction Level Parallelism)
  - Paralelismus zakódován přímo v instrukci
  - Závislosti v paralelní skupině
    - Zákaz RAW, WAW
    - Povolení WAR

srovnání: IA32 volné kódování instrukcí, 1B až 5B, strašně složitě se to dekóduje

Jednočipy

průmyslové stroje - pračka apod. :-)

- Malý paměťový prostor třeba 64kB
  - Optimalizace na velikost kódu i dat
    - Bitové proměnné
- Oddělené paměťové prostory
  - Kód a několik datových prostorů
  - Různý přístup
- Jednodušší pipeline, žádná superskalarita

mobily dřív taky 1čipy, dnes ARM (RISC, rychlej, malá spotřeba, ale ošklivý

Generování kódu

- Přidělování paměti
- Výběr instrukcí
- Přidělování registrů
- Instrukční scheduling
- Generování výsledku
  - Dohodnutý formát souboru
  - Objecty
    - Kód, data, relokace, externí a public symboly, ladící informace

Přidělování paměti

- Kód
  - Řešení skoků
- Statická oblast dat
  - Globální data
- Zásobník
  - Lokální proměnné a parametry funkcí
- Velikost datových typů
- Umístění v paměti
- Zarovnání dat `aby to četlo rychlejc, např. zarovnat na 4B, 8B…`
- Co bude v paměti a co v registrech

Výběr instrukcí

- 1:N
  - Z jedné instrukce mezikódu se generuje N instrukcí
  - Jednoduché, ale ne příliš dobrý kód
- M:N
  - Z M instrukcí mezikódu N instrukcí kódu
  - NP-úplný problém výběru
  - Heuristiky `to je to co dělá překladač dobrym nebo špatnym a je to to co výrobci překladačů tajej`
- Více cest, jak stejnou věc vygenerovat
  - CISC, SIMD, VLIW
- Problémy s neortogonální sadou instrukcí

Přidělování registrů

- Co bude v paměti a co v registrech a v jakých
  - Typy registrů
- Problémy s neortogonální sadou instrukcí
- Obarvení grafu životnosti proměnných ve funkci
  - Graf je pouze pro daný typ proměnných (např. celočíselné)
  - Počet barev je počet registrů pro daný typ
  - Pokud nejde obarvit, odstraní se uzly, které nejsou moc „důležité“ a zkusí se to znovu
  - Není třeba najít ideální obarvení největšího počtu uzlu zadaným počtem barev, ale nějaké obarvení

(= použití grafového algoritmu v praxi)

Instrukční scheduling

- Přeházení instrukcí, aby zachovávalo sémantiku a nedocházelo k zastavení pipeline
  - Závislosti RAW, WAW, WAR
  - Sestrojení grafu závislosti dat
    - Orientovaný graf, kde hrana říká, která instrukce musí předcházet jinou instrukci, aby data byla k dispozici
    - Nalezení topologického třídění
  - Latence instrukcí, spekulace, superskalarita, out-of-order execution
  - Obvykle se provádí nad jedním základním blokem