Mikroarchitektura Power 8

Maciej Buszka

POWER8 Core

Execution Improvement vs. POWER7

- SMT4 → SMT8
- 8 dispatch
- 10 issue
- 16 execution pipes:
  - 2 FXU, 2 LSU, 2 LU, 4 FPU, 2 VMX, 1 Crypto, 1 DFU, 1 CR, 1 BR
- Larger Issue queues (4 x 16-entry)
- Larger global completion, Load/Store reorder
- Improved branch prediction
- Improved unaligned storage access

Larger Caching Structures vs. POWER7

- 2x L1 data cache (64 KB)
- 2x outstanding data cache misses
- 4x translation Cache

Wider Load/Store

- 32B → 64B L2 to L1 data bus
- 2x data cache to execution dataflow

Enhanced Prefetch

- Instruction speculation awareness
- Data prefetch depth awareness
- Adaptive bandwidth awareness
- Topology awareness

Core Performance vs . POWER7

~1.6x Single Thread

~2x Max SMT

W liczbach

I-Cache	32KB
D-Cache	64KB
L2 Cache	512KB
SMT	Do 8 wątków na rdzeń
Pobieranie instrukcji	Do 8 instrukcji na cykl
Zlecanie instrukcji	Do 10 instrukcji na cykl
Zatwierdzanie instrukcji	Do 8 instrukcji na cykl

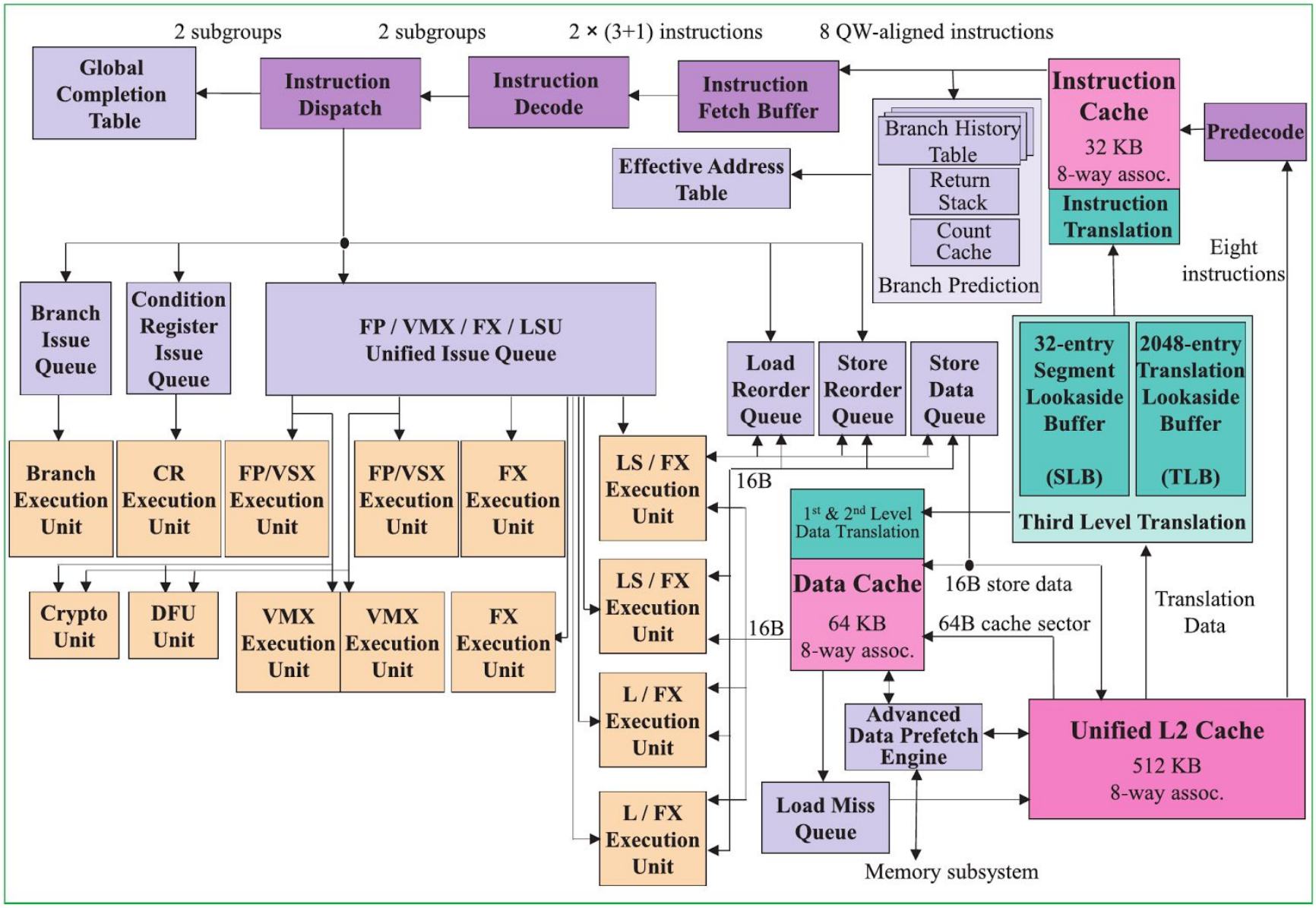

Figure 2

POWER8 processor core pipeline flow. QW-aligned refers to a quadword or 16-byte aligned address.

Organizacja rdzenia 1

- Każdy rdzeń może mieć do 8 wątków (T0, …, T7)
  - W trybie jednowątkowym dokładnie jeden z nich jest wykonywany
  - Dodatkowo są tryby 2, 4, 8 wątkowe
- Większość instrukcji (poza skokami i instrukcjami warunków) jest zlecana do Unified Issue Queue
- Unified Issue Queue składa się z dwóch symetrycznych połówek
  - Każda z nich ma podłączenie do własnych jednostek funkcyjnych:
    - FXU (FiXed point Unit)
    - VSU (Vector Scalar Unit, Floating Point Unit)
      - Implementuje: VSX, VMX
    - LU (Load)
    - LSU (Load Store)
  - Współdzielą one jednostki
    - DFU (Decimal Floating point)
    - Crypto

Organizacja rdzenia 2

- Unified Issue Queue składa się z dwóch symetrycznych połówek
  - Mają one swoją kopię pliku rejestrów
    - GPR (General Purpose Registers)
    - VSR (Vector Scalar Registers)
  - W trybie jednowątkowym
    - zawartość rejestrów jest taka sama dla obu kopii
    - instrukcje tego samego typu są zlecane na przemian do dwóch połówek
  - W trybach wielowątkowych
    - instrukcje wątków T0, T2, T4 i T6 są zlecane do UQ0
    - instrukcje wątków nieparzystych są zlecane do UQ1
    - zawartość rejestrów jest różna w UQ0 i UQ1
- Kolejki dla skoków i instrukcji warunkowych są osobne i współdzielone przez wszystkie wątki

Pobieranie i dekodowanie instrukcji (IFU)

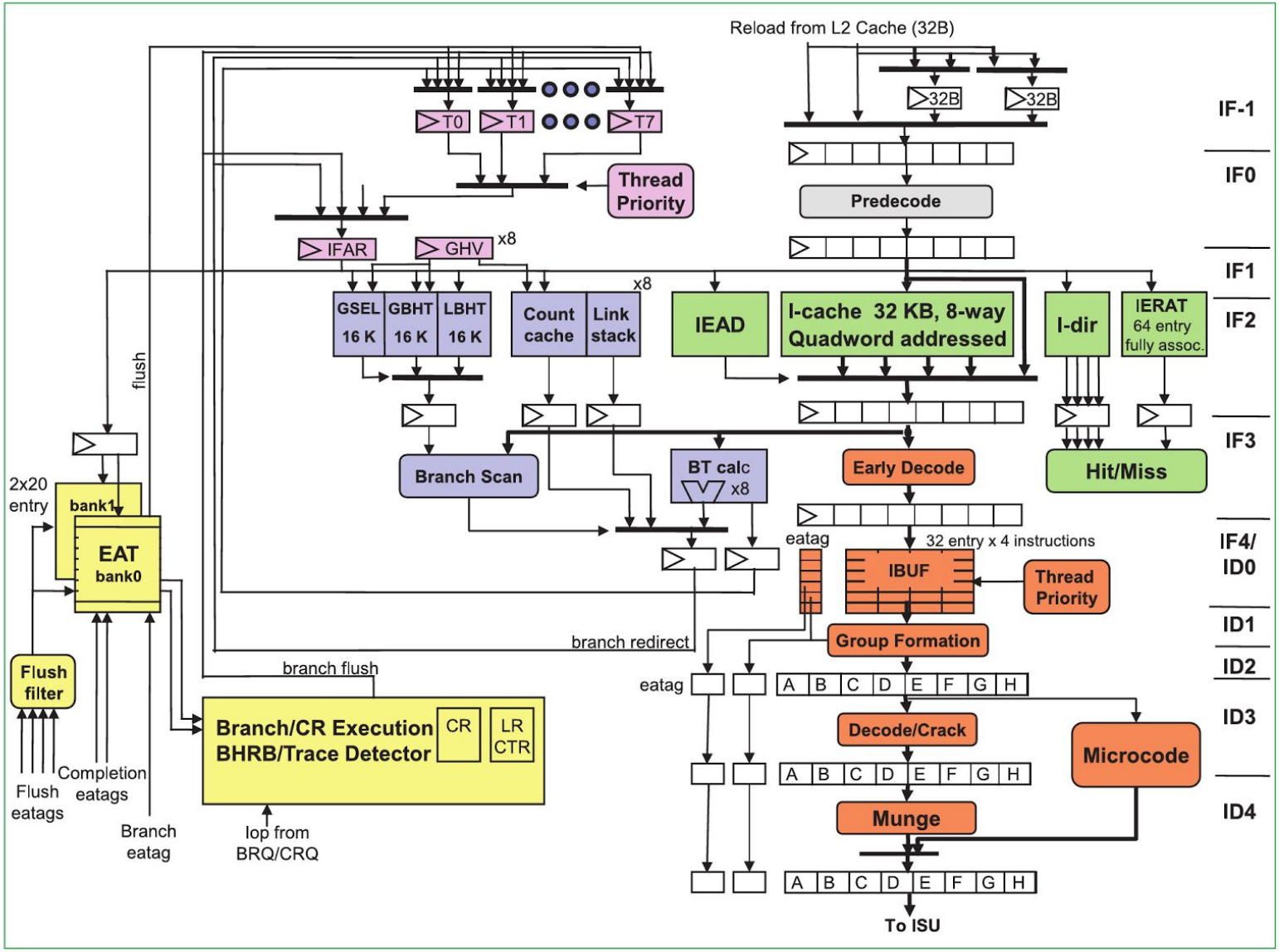

Figure 3

POWER8 instruction fetch unit logical flow. The labels on the right of the figure denote the instruction fetch (IF) and instruction decode (ID) stages. (EAT: effective address table, eatag: effective address tag; iop: internal operation.)

Instruction cache

- Cache
  - 32KB, 8 drożny
  - Długość linii - 128B, pozwala na dostęp do adresów wyrównanych do 16B
  - zaimplementowany za pomocą 16 drożnych banków aby umożliwić jednoczesne zapisy i odczyty
  - Pozwala na prefetching
  - Przy pobieraniu instrukcji z L2 wykonuje pre-decoding pozwalający na wykrycie skoków, pomagający w grupowaniu instrukcji oraz znajdowaniu wyjątków
- IEAD
  - 32 * 8 wpisów
  - Bufor adresowany efektywnym adresem pozwalający na wybór zbioru
  - Równolegle wykonywany jest dostęp do tradycyjnego katalogu opisującego cache, adresowanego fizycznie
- IERAT
  - 64 wpisy
  - Bufor adresowany efektywnym adresem, zawiera adresy fizyczne
  - W przypadku gdy adresu nie ma w buforze, trzeba wykonać normalną translację

Grupowanie i dekodowanie instrukcji

- Po wczytaniu instrukcje lądują w buforach instrukcji
  - 32 wpisy po 4 instrukcje
- Następnie logika wyboru wątku tworzy grupę składającą się z:
  - W przypadku jednowątkowym
    - do 6 instrukcji nie będących skokami
    - nie więcej niż 2 skoków
  - W przypadku wielowątkowym
    - do 3 instrukcji nie będących skokami
    - nie więcej niż 1 skoku
- Taka grupa jest przekazywana do dekodowania/rozbijania
  - Proste instrukcje są dekodowane
  - Złożone instrukcje wymagające do trzech prostych operacji są bezpośrednio rozbijane
  - Pozostałe złożone instrukcje są przekazywane do silnika mikrokodu, które wytworzy instrukcje symulujące ich działanie
- Dodatkowo dodawane są różne markery pomagające przy odpluskwianiu, czy też monitorowaniu wydajności

Fuzja instrukcji

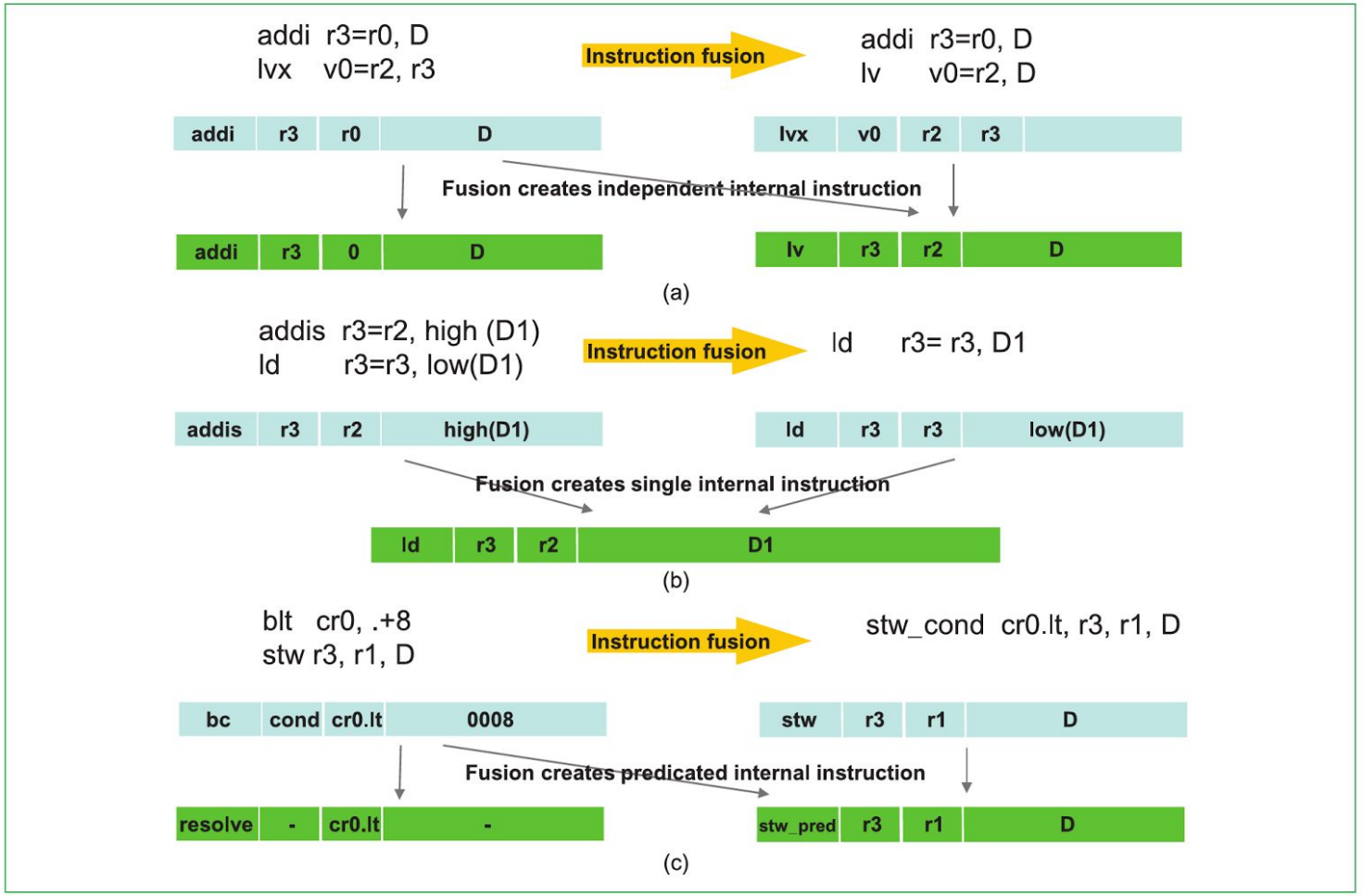

Figure 4

Instruction fusion in the POWER8 processor can be used for a variety of purposes that improve performance. (a) Two dependent instructions are transformed into two independent internal operations. (b) Two dependent instructions are transformed into a single internal operation. (c) A branch is transformed into predicated execution.

Predykcja skoków

- Po pobraniu instrukcji procesor skanuje je w poszukiwaniu skoków, sprawdza ich predykcje i wpisuje przewidziany adres docelowy pierwszego skoku przewidzianego jako wykonany
  - Zajmuje to 3 cykle jeżeli wśród instrukcji znajduje się skok, w czasie których dany wątek nie może pobrać kolejnych instrukcji
  - Przy braku skoków procesor może kontynuować pobieranie instrukcji
- W architekturze Power są dwa typy skoków:
  - branch
    - addr := PC + offset
    - adres docelowy może zostać obliczony podczas pobierania instrukcji, z danych które są w niej zakodowane
  - branch to link / branch to count
    - addr := LinkReg / CountReg
    - adres docelowy zależy od rejestru, który może się zmieniać do momentu wykonania tej instrukcji
- W związku z tym, predykcja skoków rozdzielona jest na dwie części:
  - Predykcja kierunku (wykonany / niewykonany)
  - Predykcja adresu (dla instrukcji branch to link/count)

Predykcja kierunku

- Tablice historii
  - Współdzielone między wszystkie wątki
  - Wpisy składają się z bitu kierunku oraz bitu histerezy
- Lokalna tablica historii (LBHT)
  - 16K wpisów
  - Adresowana ostatnimi 14 bitami adresu instrukcji
- Wektor historii (GHV)
  - 21 bitów
  - Unikalny dla każdego wątku
- Globalna tablica historii (GBHT)
  - 16K wpisów
  - Adresowana hashem adresu i GHV
- Globalna tablica wyboru (GSEL)
  - 16K wpisów
  - Adresowana hashem adresu i GHV
  - Wybiera z której predykcji należy korzystać

Predykcja adresu (skoki niebędące powrotami)

- Local count cache
  - 256 wpisów
  - indeksowana 8 bitami adresu instrukcji
  - każdy wpis zawiera:
    - 62 bity adresu
    - bit pewności
    - 2 bitowy licznik nasycający, wskazujący czy korzystać z cache lokalnego czy globalnego
- Global count cache
  - 512 wpisów
  - indeksowana XORem 9 bitów adresu instrukcji i 9 bitów GHV
  - każdy wpis zawiera:
    - 30 dolnych bitów adresu (górne są przepisywane z aktualnego PC)
    - 2 bity pewności

Predykcja adresu (powroty z procedur)

- Każdy wątek posiada stos połączeń (link stack)
- Za każdym razem gdy wśród pobranych instrukcji znajduje się instrukcja branch-and-link adres następnej instrukcji jest odkładany na stos
- Natomiast przy każdej instrukcji branch-to-link wierzchołek stosu jest usuwany (i staje się przewidywanym adresem)
- Oczywiście te operacje wykonywane są spekulatywnie
- Pojemność stosu (dla każdego wątku):
  - ST oraz SMT2 - 32 wpisy
  - SMT4 - 16 wpisów
  - SMT8 - 8 wpisów

Wykrywanie hazardów SHL

- SHL - Store-Hit-Load
- Hazard występujący w potoku przetwarzania poza porządkiem programu
- Występuje gdy starszy zapis wykonuje się później niż młodszy odczyt z danego adresu
- W przypadku gdy IFU wykryje taki hazard instrukcja odczytu zostanie usunięta i pobrana na nowo oraz oznaczona
- Dla tak oznaczonych odczytów w kolejnych częściach potoku zostanie utworzona zależność danych od konfliktującego zapisu

Dynamiczna konfiguracja IFU

- Mikroarchitektura Power8 udostępnia specjalny rejestr pozwalający na konfigurację różnych części IFU
- Funkcjonalność udostępniona jest jako rejestr WORT (Workload Optimization Register for Thread control)
- Można ustawić między innymi:
  - Użycie globalnych predyktorów skoku
  - Prefetching I-Cache
  - Spekulację
  - Fuzje instrukcji (BC + _) do instrukcji z predykatem
  - Unikanie SHL
  - Kończenie grupy instrukcji na skoku wstecz

Sekwencjonowanie instrukcji (ISU)

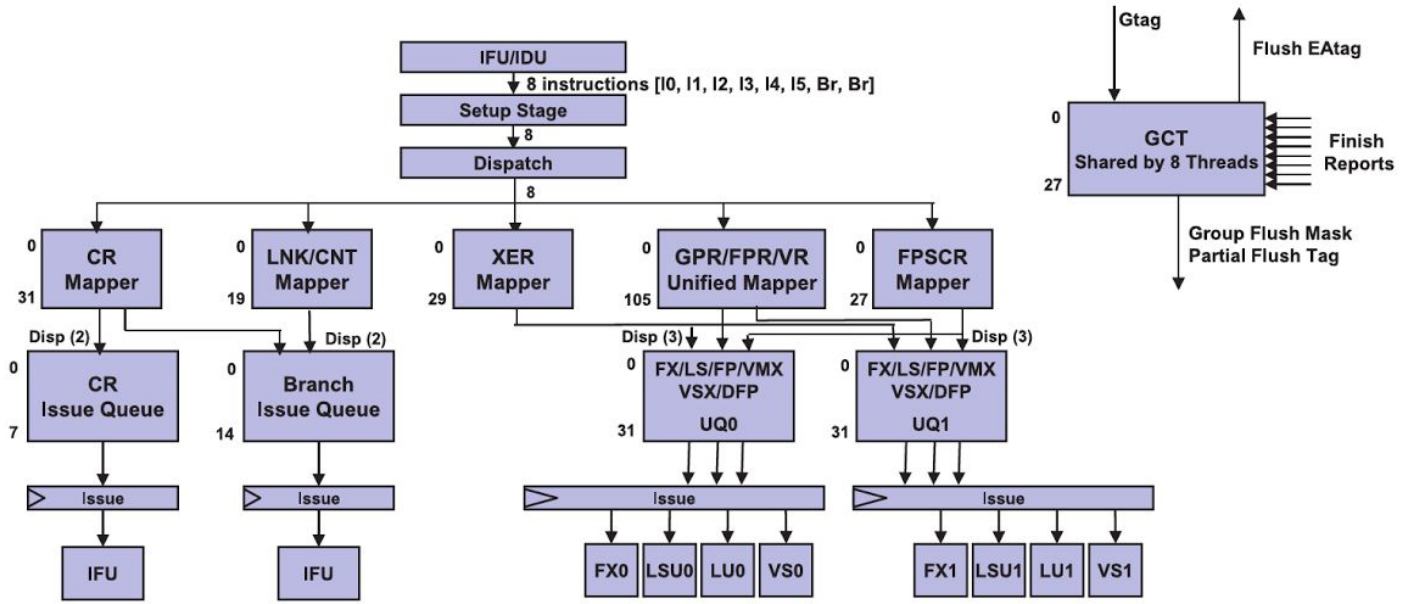

Figure 5

POWER8 instruction sequencing unit (logical flow).

Planowanie instrukcji

- Instrukcje planowane (schedule) są grupami
  - 6 instrukcji zwykłych + 2 skoki w trybie ST
  - 2 * (3 instrukcje zwykłe + 1 skok) w trybie SMT
    - grupy mogą pochodzić od różnych wątków
- Wszystkie zasoby muszą być dostępne dla każdej instrukcji w grupie aby mogła ona zostać zaplanowana
  - W przeciwnym przypadku zostanie ona zatrzymana przed planowaniem
- Dostępne kolejki
  - Branch Queue (BQ)
    - 15 pozycji
    - kolejka przesuwna
  - Condition Register Queue (CRQ)
    - 8 pozycji
    - kolejka przesuwna
  - Unified Queue (UQ0, UQ1)
    - 2 * 32 wpisy
    - kolejka nieprzesuwna (początek i koniec zarządzany przez wskaźniki)
    - relatywny wiek instrukcji śledzony jest w specjalnej macierzy

Zlecanie do wykonania

- Instrukcje z każdej z kolejek mogą zostać zlecone do wykonania poza porządkiem programu
- W jednym cyklu można zlecić
  - jedną instrukcję skoku
  - jedną instrukcję logiczną
  - Po jednej instrukcji do każdej jednostki funkcyjnej obu połów UQ
  - Razem do 10 instrukcji na cykl
- Tagi operacji na pamięci
  - Operacje na pamięci mają dodatkowe bufory i tagi będące wskaźnikami na ich elementy
    - LRQ - Load Reorder Queue - LTAG
    - SRQ - Store Reorder Queue - STAG
  - Aby zapobiec wstrzymywaniu planowania, ISU korzysta z wirtualnych tagów, które zamieniane są na fizyczne gdy zwolni się miejsce w kolejce (2 * 128 tagów)
  - Operacja może zostać zlecona, gdy ma przypisany fizyczny tag
    - instrukcja zapisu jest zlecana dwukrotnie z takim samym tagiem:
      - raz jako store data op. - co powoduje przekopiowanie danych do zapisu
      - raz jako store address op. - aby obliczyć adres do zapisu

Zatwierdzanie instrukcji

- Global Completion Table
  - 28 wpisów
  - Każdy wpis to jedna grupa do 8 instrukcji (ST) lub dwie grupy po 4 instrukcje z tego samego wątku (SMT)
  - Łącznie daje to możliwość utrzymania maksymalnie 224 instrukcji w locie
  - Każdy wpis zawiera bity zakończenia dla każdej instrukcji
  - Każda instrukcja wykonywana jest poza porządkiem programu
  - Zatwierdzone mogą zostać albo dwie grupy różnych zbiorów wątków (parzystych i nieparzystych) w przypadku SMT, albo jedna grupa (ST)
  - Identyfikator grupy (GTAG)
    - Udostępniany podczas zatwierdzania lub unieważniania grupy
- Unieważnianie instrukcji
  - Unieważnienia całych grup są zbierane w 28 bitową maskę
  - Częściowe unieważnienie w obrębie jednej grupy za pomocą GTAG i 6 bitowej maski
    - Jako, że częściowe unieważnienie będzie wynikiem złej spekulacji pierwszego skoku, a potencjalny drugi skok będzie ostatnią instrukcją w grupie

Przemianowanie rejestrów (GPR)

- GPR (General Purpose Registers)
  - Dwa pliki po 124 wpisy
    - (96 rename, 32 architected) dla ST, oba pliki mają taką samą zawartość
    - (96 rename, 32 architected) dla SMT2, pliki mają różną zawartość
    - (60 rename, 64 architected) dla SMT4, pliki mają różną zawartość
    - (60 rename, 64 architected) dla SMT8, pliki mają różną zawartość
      - nie wszystko się tutaj mieści, pozostałe rejestry są w SAR
- SAR (Software Architected Registers)
  - Po jednym pliku na wątek, 72 wpisy
    - 32 architected + 4 eGPR (dla instrukcji z mikrokodu)
    - 32 + 4 kopia dla pamięci transakcyjnej
- ISU śledzi w którym z plików znajduje się aktualny rejestr architektoniczny

Przemianowanie rejestrów (VSR) oraz pozostałe

- VSR (Vector Scalar Registers)
  - Dwa pliki po 144 wpisy
    - (80 rename, 64 architected) dla ST, oba pliki mają taką samą zawartość
    - (80 rename, 64 architected) dla SMT2, pliki mają różną zawartość
    - (60 rename, 64 architected) dla SMT4/8, pliki mają różną zawartość
      - nie wszystko się tutaj mieści, pozostałe rejestry są w SAR
- SAR
  - Po jednym pliku na wątek, 124 wpisy
    - 64 architected
    - 64 kopia dla pamięci transakcyjnej
- Pozostałe rejestry które także mogą zostać przemianowane:
  - CR - 32 rename + 8 * 8 architected
  - LR, CTR, TAR - 20 rename + 8 * 3 architected
  - XER - 30 rename + 8 * 4 architected + 8 architected (bez przemianowania)

Podsystem pamięci (LSU)

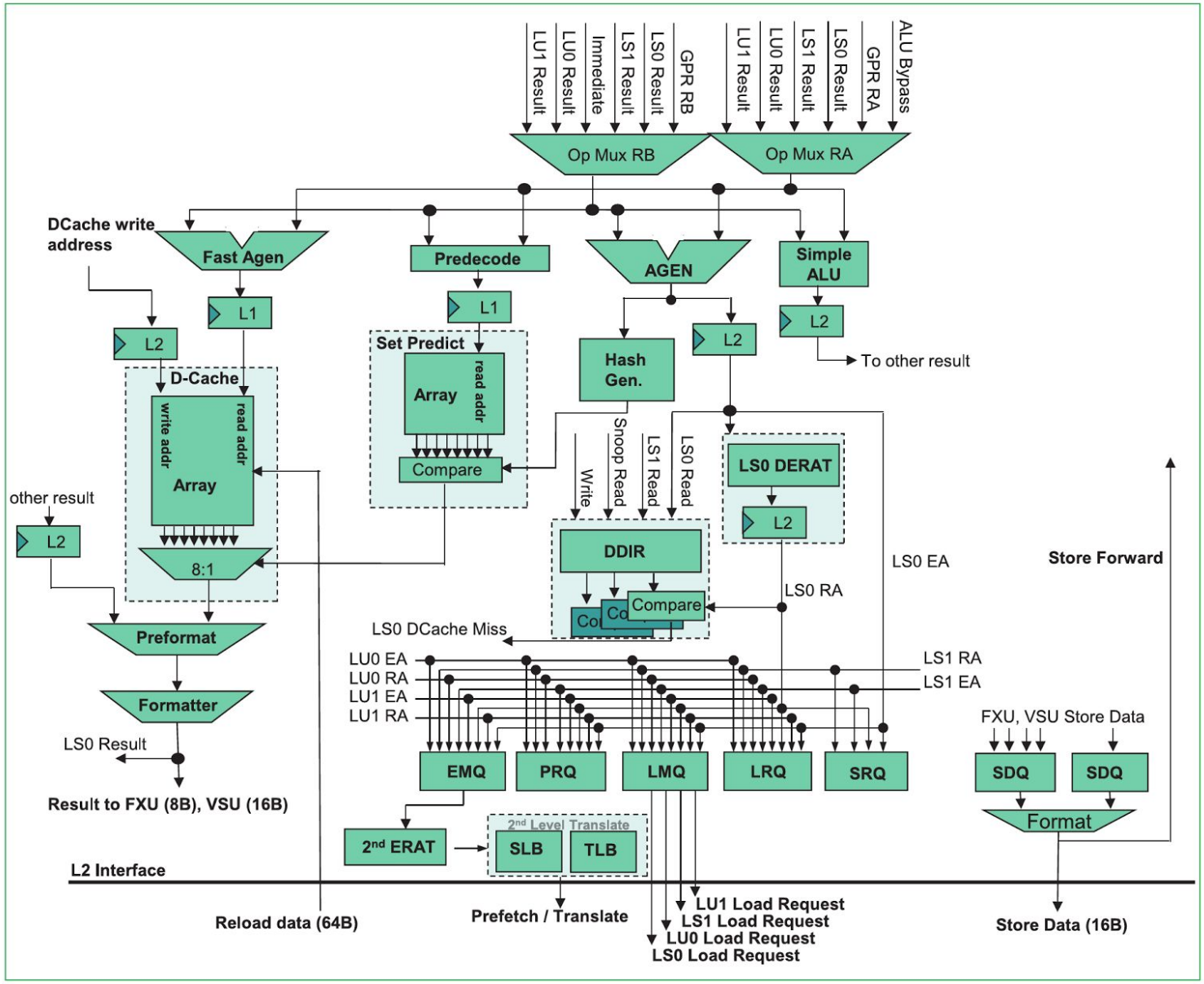

Figure 6

POWER8 LSU micro architecture (LS0 pipe shown).

Interfejs podsystemu pamięci

- W każdym cyklu można zlecić po jednej instrukcji do pary LS0/L0 i LS1/L1
  - instrukcje zlecane są poza porządkiem programu
- Jednostki te potrafią także wykonywać proste operacje fixed-point
  - czas trwania - 3 cykle
- Wykonanie instrukcji load przy trafieniu w D-Cache
  - 3 cykle dla wczytania 8B fixed-point
  - 5 cykli dla wczytania 16B vector
- Instrukcja store zlecana jest dwa razy
  - generowanie adresu do LS
  - wczytanie danych do L

Zachowywanie porządku

- Kolejka przestawiania zapisów (SRQ)
  - 40 wpisów, tablica asocjacyjna indeksowana adresami fizycznymi
    - alokowane przy zlecaniu instrukcji, dealokowane po zapisaniu do cache
  - Każdemu z wpisów odpowiada element kolejki przestawiania danych (SDQ) 16B
- Kolejka przestawiania odczytów (LRQ)
  - 44 wpisy, tablica asocjacyjna indeksowana adresami fizycznymi
  - Monitoruje hazardy
    - Na przykład młodsza instrukcja wykona się przed starszym zapisem (SHL) lub odczytem
    - W przypadku hazardu unieważnia instrukcje zależne od tego odczytu, po czym wymusza ponowne pobranie instrukcji

Translacja adresów 1

- 64 bitowe adresy efektywne
- 50 bitowe adresy fizyczne
- Strony wielkości 4K, 64K, 16M, 16G
- Translacja 1 poziomu
  - DERAT - tabela efektywny -> fizyczny
    - Primary
      - 48 wpisów, w pełni asocjacyjny
      - 4 kopie, synchronizowane parami (L - LS)
      - strony 16G są rozbijane na strony 16M
      - binary tree LRU
    - Secondary
      - 256 wpisów, w pełni asocjacyjny
      - w trybie SMT traktowany jako dwie tablice 128 wpisowe (osobne dla różnych zbiorów wątków)
      - strony
      - FIFO

Translacja adresów 2

- 64 bitowe adresy efektywne
- 78 bitowe adresy wirtualne
- 50 bitowe adresy fizyczne
- Translacja 2 poziomu
  - Segment Lookaside Buffer
    - Adres efektywny -> wirtualny
    - Kopia dla każdego wątku
    - Segmenty wielkości 256M lub 1T
    - Wiele wielkości stron na segment - MPSS
    - 32 wpisy, w pełni asocjacyjny
    - Zarządzany w trybie nadzorcy
  - Translation Lookaside Buffer
    - Adres wirtualny -> fizyczny
    - 2048 wpisów, 4 drożny
    - Współdzielony między wszystkie wątki
    - True LRU
    - Maksymalnie 4 page-walk jednocześnie
    - Hit under Miss - Trafienia mogą być wykonywane w trakcie page-walk

Translacja adresów 3

- Translacja 2 poziomu
  - Segment Lookaside Buffer
    - Adres efektywny -> wirtualny
    - Kopia dla każdego wątku
    - Segmenty wielkości 256M lub 1T
    - Wiele wielkości stron na segment - MPSS
    - 32 wpisy, w pełni asocjacyjny
    - Zarządzany w trybie nadzorcy
  - Translation Lookaside Buffer
    - Adres wirtualny -> fizyczny
    - 2048 wpisów, 4 drożny
    - Współdzielony między wszystkie wątki
    - True LRU
    - Maksymalnie 4 page-walk jednocześnie
    - Hit under Miss - Trafienia mogą być wykonywane w trakcie page-walk
    - Każdy wpis oznaczony jest LPAR (Logical Partition)
      - Aby zarejestrować trafienie LPAR musi zgadzać się LPAR aktywnej partycji
      - Dzięki temu przy zmianie partycji nie trzeba unieważniać starych wpisów

Data cache

- 64K, 8 drożna
  - 4 porty odczytu, 1 zapisu
  - Długość linii - 128B, podzielone na 32B sektory
  - Hybrid LRU
- Zapisy
  - Wyższy priorytet niż odczyty
  - Store-through, strictly inclusive
  - W przypadku cache-miss przy zapisie linia nie zostanie wczytana do L1
  - Zapisy do 16B
- Odczyty
  - czas dostępu 3 cykle, o ile nie przecinają linii cache

Data cache implementacja

- Zaimplementowana jako 16 makr, 16 banków każde
  - Jeden bank może wykonać dwa odczyty lub jeden zapis na raz
  - Makro udostępnia dwa odczyty i jednoczesny zapis (do różnych banków)
  - W przypadku konfliktu odczyty zostaną odrzucone i zlecone ponownie
- Komunikacja z L2
  - Store do 16B
  - Line reload 32/64B
  - Walidacja linii na poziomie 32B sektorów
  - Nowa linia jest domyślnie współdzielona między wszystkie wątki
- Predykcja zbioru
  - 64 * 8, 8 drożna
  - Indeksowana 6 bitami adresu (tymi samymi co D-Cache)
  - Jednocześnie część adresu jest haszowana i porównywana z zawartością zbioru, oraz sprawdzane są tagi wątku
  - W przypadku trafienia zawartość odpowiadającego zbioru w cache jest potencjalnym trafieniem
  - Jednocześnie pobierany jest pełen adres fizyczny z katalogu i porównywany z adresem przetłumaczonym przez DERAT

Obsługa cache-miss

- Kolejka nietrafionych odczytów (LMQ)
  - 16 wpisów
  - Nietrafiony odczyt zwalnia wpis w kolejce zlecania i tworzy wpis w LMQ
    - Jednocześnie zleca cache-line reload
    - Gdy dane pojawią się w D-Cache ma on najwyższy priorytet i dane są transferowane do docelowego rejestru
  - Dostęp na podstawie adresu fizycznego
  - Współdzielona między wątki
  - Obsługuje do ośmiu jednoczesnych operacji na każdy zbiór cache

Data prefetch

- Zoptymalizowany pod sekwencyjne (rosnące bądź malejące) dostępy do pamięci
- Potrafi wykryć ciągi dostępów do stycznych linii cache
- Wykonuje prefetch na wszystkich poziomach (L1, L2, L3)
- Na początku wczytuje wiele linii, potem redukuje ich ilość tak aby osiągnąć ich optymalną ilość
- Rozpoczyna się przy D-Cache miss, korzysta z 16 pozycyjnego bufora
- Ciągi linii mogą przekraczać granicę 4K i 64K strony
    - A zatem wszystkie adresy muszą przejść translację

Jednostki obliczeniowe (FXU, VSU)

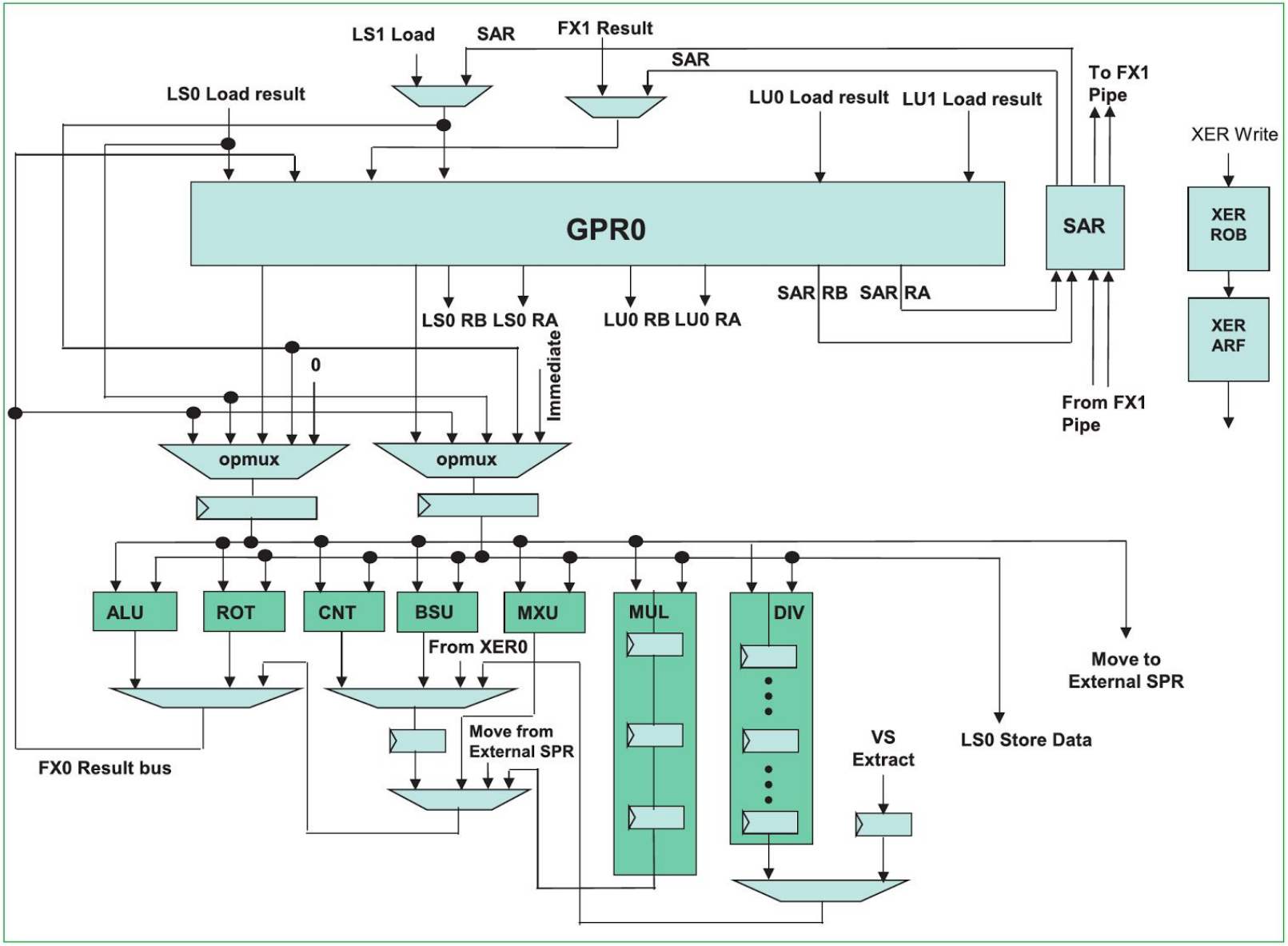

Figure 7

POWER8 FXU overview (FX0 pipe shown).

Jednostka FXU

- Plik rejestrów ma 8 portów odczytu i 6 portów zapisu
  - W trybie ST porty zapisu (LS oraz FX) podłączone do odległych jednostek (z drugiej połowy UQ) są współdzielone z zapisem do SAR
- Dla prostych operacji, czas wykonania to 1 cykl
  - tzn, że zależne operacje mogą być wykonywane jedna po drugiej, o ile pochodzą z tej samej połowy UQ
  - Pozostałe czasy to 2, 4 lub zmienna ilość cykli
- VSU extract bus
  - pozwala na szybkie przenoszenie danych z rejestrów wektorowych
- Dzielenie wykorzystuje algorytm SRT (radix = 16)
  - Podczas wykonywania dzielenia inne operacje mogą być zlecane do pozostałych części FXU
- XER - Rejestr wyjątków

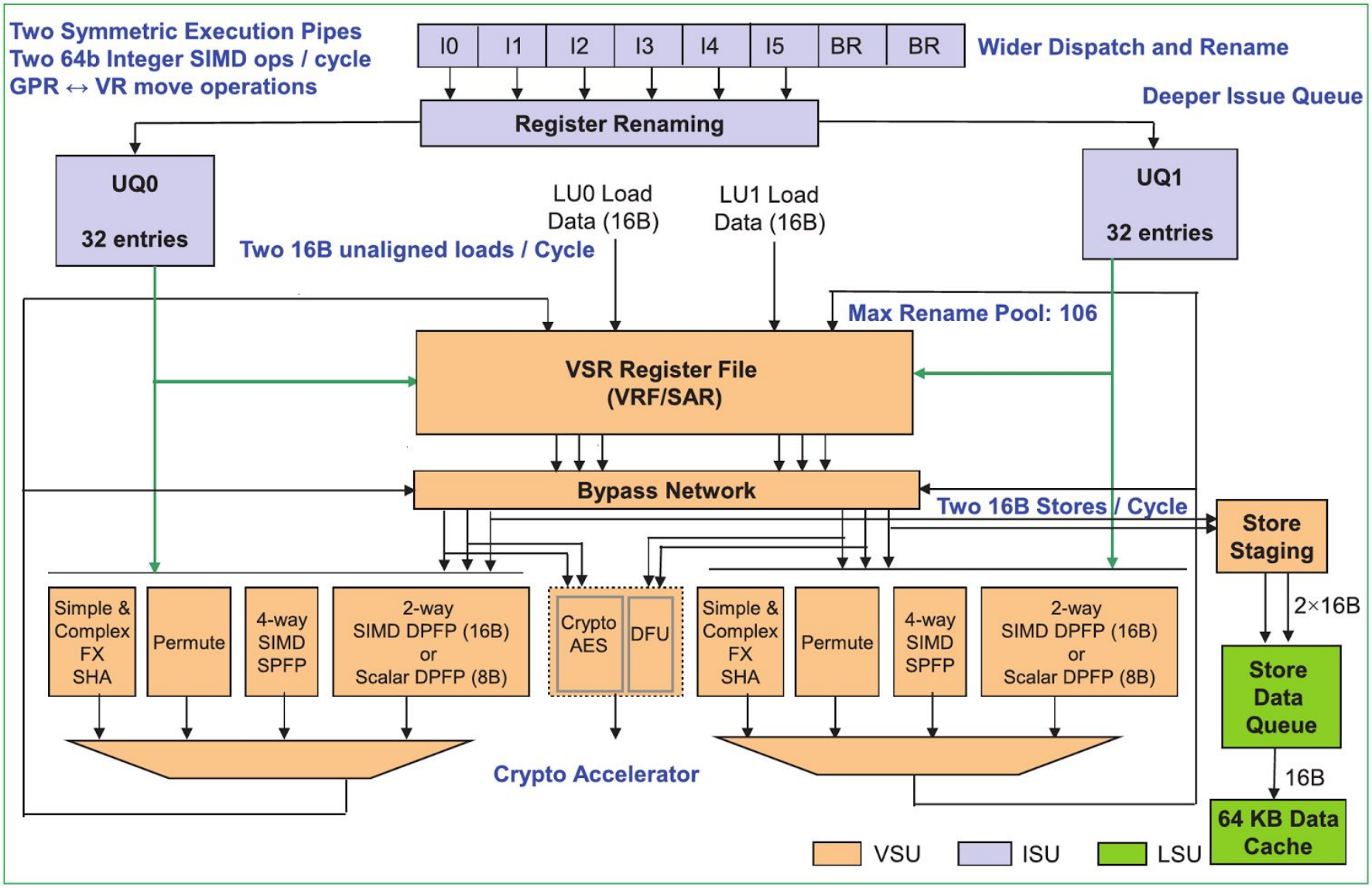

Figure 8

POWER8 fully symmetric VSU pipelines. (SPFP: single-precision floating-point; DPFP: double-precision floating-point.)

Jednostka VSU

- Implementuje operacje
  - zmiennoprzecinkowe
    - 2 * 32-bit lub 1 * 64-bit
  - wektorowe skalarne
    - do 4 32-bitowych mnożeń na cykl (potok)
  - wektorowe zmiennoprzecinkowe
    - do 4 operacji 64-bit (2 multiply-add) na cykl
- Load/Store 32B na cykl
- Data bypassing
- VMX Crypto - AES, SHA2, CRC

Bibliografia

- IBM POWER8 processor core microarchitecture
- Power ISA v.2.07