Wojciech Zatorski, lab 7, gr. 38, rok 3.

SPRZĘT KOMPUTEROWY

Sprawozdanie nr 1

1. Symulacja potoku – przetwarzanie potokowe

Przetwarzanie potokowe to inaczej praca potokowa lub pipelining - równoległe wykonywanie w procesorze poszczególnych faz cyklu rozkazowego. Jeden z układów procesora potokowego wykonuje rozkaz, a drugi, niezależny układ w tym samym czasie pobiera następny rozkaz z pamięci, co podwaja wydajność procesora. Stopień równoległości konstrukcji w architekturze RISC wynosi od trzech do siedmiu niezależnych bloków funkcjonalnych wykonujących jednocześnie pobieranie rozkazu, jego zdekodowanie, obliczanie adresu argumentu, pobranie argumentu, wykonanie operacji i przechowywanie wyniku. Przetwarzanie potokowe stosowane są we wszystkich współczesnych procesorach. Dzięki niemu można szybko wykonywać kilka instrukcji, przy czym w danej chwili każda będzie w innej fazie wykonywania.
Potokowa architektura procesora jest bardzo wydajna, gdy nie występują przerwania, instrukcje warunkowe oraz przede wszystkim skoki, bowiem jest ona bardzo wrażliwa na nie. Jeżeli taka jednostka trafi na rozkaz skoku, wówczas pierwsze stopnie potoku muszą zostać opróżnione, co powoduje utratę co najmniej kilku cykli zegarowych, co jest równoważne wykonaniu kilku instrukcji. Jednakże zostały skonstruowane specjalne jednostki wykrywające skoki, które minimalizują skutki wypróżniania potoków. Układy te przewidują adres następnej instrukcji już w pierwszej fazie jej wykonywania (na podstawie adresu instrukcji). Jeśli układ przewidywania skoków dobrze je odgadnie, to zaoszczędzi czas równoważny wykonaniu kilku instrukcji.

	bez skoków	2 skoki na początku	2 skoki pod koniec
1	A	A	A
2	B A	B A	B A
3	C B A	**S** B A	**S** B A
4	D C B A	E	C **S** B A
5	E D C B A	F E	D C **S** B A
6	F E D C B	G F E	E D C **S** B
7	G F E D C	H G F E	H
8	H G F E D	I H G F E	I H
9	I H G F E	J I H G F	J I H
10	J I H G F	K J I H G	K J I H
11	K J I H G	L K J I H	L K J I H
12	L K J I H	M L K J I	M L K J I
13	L K J I	N M L K J	N M L K J
14	L K J	**S** N M L K	**S** N M L K
15	L K	O	O **S** M L K
16	L		P O **S** M L
	Wykonanych 12 rozkazów	Wykonanych 6 rozkazów	Wykonanych 7 rozkazów

Wnioski:

Przetwarzanie potokowe daje duży przyrost wydajności procesora, jeśli nie występują skoki, w przypadku występowania skoków wydajność spada bardzo drastycznie. Im więcej skoków tym gorsze rezultaty. Podwyższenie sprawności przetwarzania potokowego daje zastosowanie mechanizmów wykrywających skoki, im bardziej dokładny mechanizm wykrywania tym znacznie wzrasta wydajność procesora (do początku, gdy nie było skoków).

Przy przetwarzaniu potokowym na każdy cykl powinien przypadać jeden wykonany rozkaz ,a w przypadku przetwarzania super skalarnego, kilka. Średnia liczba wykonanych rozkazów w cyklu nazywa się wydajnością (performance). Nominalna wartość zależy od struktury potoku, ilości podetapów. Rzeczywista wartość ze względu na konflikty danych, sterowania i dostępu jest zawsze mniejsza od nominalnej. Dla dowolnego potoku można ją określić jako odwrotność średniej liczby cykli przypadających na wykonanie rozkazu.

P=p((1-b-r)+r(1+d)+bn)-1 [rozkaz/cykl]
p - nominalna wydajność potoku
n - głębokość potoku
b - częstotliwość nieprzewidzianych skoków opóźniających potok o n cykli
d - liczba opóźnionych cykli z powodu konfliktu danych
r - częstotliwość konfliktów danych

2. Klasyfikacja procesorów i architektur.

Architektura **CISC** (Complex Instruction Set Computing) to architektura konwencjonalnych procesorów. Charakteryzuje się ona znaczną liczbą elementarnych rozkazów i trybów adresowania przy niewielkiej liczbie rejestrów uniwersalnych. Jest ona bardzo wolna w porównaniu do RISC-ów. Na tej architekturze były oparte pierwsze procesory z rodziny x86. Dziś w tej rodzinie nie spotka się procesora CISC, ale jest ona zastąpiona przez wewnętrzy RISC. Jednak pozostała zgodnie z zasadą "kompatybilności w tył" mała liczba rejestrów, co jest główną bolączką projektantów procesorów x86.

Architektura **RISC** (Reduced Instruction Set Computing/Computer) wywodzi się z Berkeley (1985). Koncepcja tego typu architektury jest oparta na ograniczeniu liczby krótkich (maksymalnie dwusłowowych) rozkazów mających niewiele formatów i trybów adresowania. Procesor RISC posiada wiele rejestrów uniwersalnych (nawet powyżej 100). Działanie procesora przyśpieszają dodatkowe ulepszenia, np. przetwarzanie potokowe, czy pamięć podręczna. Instrukcje wykonują proste operacje, dzięki czemu mogą to robić szybko - każda jednostka wykonawcza jest w stanie wykonać dokładnie jedną instrukcję w jednym cyklu zegara. Lista instrukcji zawiera tylko kilka rozkazów umożliwiających dostęp do pamięci: załadowanie (load), zapis (store) oraz instrukcje semaforowe. Wszystkie pozostałe rozkazy operują wyłącznie na rejestrach - to również upraszcza budowę układu. Instrukcje kodowane są w prosty sposób - każdy rozkaz ma taką samą długość (np. 32 bity). Mimo swych zalet ta architektura ma kilka wad, a najważniejszą jest znaczna komplikacja procesora przy zastosowaniu np. nie kolejnego wybierania instrukcji i innych elementów superskalarnych. Przyczyną jest niejawna równoległość przetwarzania danych. Złożone jednostki analizują relacje między poszczególnymi operacjami, co znacznie powiększa powierzchnię procesora, bo jeżeli np. dwie instrukcje korzystają z tego samego rejestru, to nie mogą zostać wykonane w tym samym czasie. Podsumowując architektura RISC jest wydajna, ale pociąga za sobą pewne komplikacje budowy.

Architektura **EPIC** - Explicitly Paralell Instruction Coumputing - to zupełnie nowa architektura opracowana przez Intela wspólnie z Hewlett-Packard. Jest ona dość młoda. Architektura ta ma zalety procesorów VLIW, ale w odróżnieniu od niej program napisany na starszy procesor będzie działał poprawnie także na nowym procesorze o większej ilości jednostek

wykonawczych. Kolejne wersje procesorów EPIC będą mogły być dowolnie rozbudowywane np. o następne równolegle pracujące jednostki wykonawcze, a programy przeznaczone dla starszych układów nadal będą działały. Istnieją jednak pewne obawy, że starszy kod nie będzie wykonywany optymalnie na nowszych strukturalnie jednostkach.

Producent	Nazwa	Architektura
Motorola	MC 68000-68060	CISC
Motorola	PowerPC	RISC
AMD	K6, K7	RISC/CISC
Intel	80x86	CISC
Intel	Pentium, Celeron	RISC/CISC
Intel	Pentium 4	RISC/EPIC
VIA	Cyrix III	RISC/CISC

3. Klasyfikacja komputerów

Architektura Von Neumana:

Podstawowa zasada funkcjonowania komputera, którym posługujemy się w szkole lub w domu nie zmieniła się zasadniczo od ponad sześćdziesięciu lat. W 1944 roku wybitny amerykański matematyk **John von Neuman** sformułował założenia funkcjonowania współczesnego komputera. Brzmią one następująco:

1. Komputer powinien być wyposażony w magazyn do przechowywania danych zwany pamięcią. W pamięci zapisywane są dane zakodowane w postaci ciągu zer i jedynek.
2. Wszystkie działania arytmetyczne, logiczne i inne są wykonywane przez specjalny element zwany arytmometrem.
3. Komputer musi mieć możliwość komunikowania się ze światem zewnętrznym czyli pobierania danych i wyprowadzania wyników.
4. Komputer w czasie swej pracy realizuje ściśle algorytm zapisany przez człowieka. Algorytm ten zapisany jest przez programistę w postaci programu komputerowego zawierającego ciąg elementarnych instrukcji postępowania.

Przykłady: UNIVAC, IBM 650, IAS.

Architektura Harwardzka:

Architektura harwardzka opiera się na użyciu dwóch oddzielnych szyn dla danych i rozkazów, dzięki czemu w trakcie pobierania argumentów wykonywanej właśnie instrukcji można równocześnie zacząć pobieranie następnego słowa rozkazowego (pre-fetch). Skraca to cykl rozkazowy i zwiększa szybkość pracy. Obszary adresowe pamięci danych i programu(wewnętrznych i czasami zewnętrznych) są rozdzielone. Pociąga to za sobą niejednoznaczność adresów, ponieważ pod tym samym adresem widzi pamięć RAM i ROM. W tym przypadku stosuje się inne rozkazy dla pamięci programu i inne dla pamięci danych. Ponadto magistrala danych i rozkazów mają różną szerokość (długość słowa), np. PIC16F87x – magistrala danych 8-bitowa, magistrala rozkazów 14-bitowa. Mapa pamięci dla architektury harwardzkiej. Wadą tego rozwiązania jest utrudniony przepływ danych z pamięci programu do obszaru pamięci operacyjnej, co uniemożliwia stosowanie jednej z podstawowych technik programistycznych (look-up tables). Innymi słowy nie jest możliwe indeksowane przesłanie danych z pamięci ROM do RAM, co oznacza np. brak możliwości budowy tabel współczynników stałych w pamięci ROM. Jedynym sposobem wbudowania stałych w program jest ukrycie ich w kodach rozkazów.

Przykłady: ENIAC, ALPHA.