1. Да се разгледа конвейрното изпълнение на програма при наличие на обща и отделни памети за данни и програми.

a) load A, M1

load B, M2

SUB A, B

store M3, A

load C, M3

add A, C

store M4, A

Оценете времето за изпълнение на фрагмента без и с конвейр.

A, B и C са регистри, M1, M2, M3 и M4 са клетки от паметта .

Инструкции зададени в задачата:

1. Зареди в регистър А съдържанието на клетка М1- (извличане)
2. Зареди в регистър В съдържанието на клетка М2 – (извличане)
3. От съдържанието на регистър А извади съдърванието на регистър В
4. Запиши в клетка М3 стойноста на регистър А
5. Зареди в регистър С стойноста на клетката М3
6. Събери регистър А с регистър С /резултата е в А/
7. Запиши в М4 стойноста на регистър А
8.

С математически означения задачата би изглеждала във вида: (x-y).2

При изпълнението на ADD и SUB са необходими две фази:

1.”I” - избор на командата;

2.”E” - изпълнение на командата.

За изпълнението на операциите за извличане/запис в/от паметта (LOAD и STORE) са необходими три фази:

1.”I” - избор на командата;

2.”E” - изчисление на адреса на паметта;

3.”D” - обръщение към паметта.

При изпълнение, зададения програмен фрафмент без конвеир, би изглеждало по следният начин:

Изчислителния процес в този случай отнема 19 такта.

Конвейрното изпълнение може да се разгледа в два варианта:

Вариант 1

При наличие на обща памет за данните и командите в процесора.

В този случай различните фази от нееднотипни команди, не могат да се стартират едновременно в един и същ такт, защото се прави опит за заемане на един и същ ресурс (паметта).Естествния ред на изпълнение на инструкциите е последователния – по принципа на Фон Нойман. Това се променя ако има циклична операция, проверка на условие, или ако тази промяна е зададена в специален операнд на инструкцията. Това е възможно защото инструкциите имат един операнд указваш адреса на следващата инструкция.

Операндите могат да бъдат от следние типове – адреси, числа, символи, логически данни и специализирани типове данни или структури данни.

Инструкциите се изпълняват по следния начин

- извлича се команда от ОП
- дешифрира се КОП
- извлича се операнда

- изпълняват се поредната операция

- запазва се резултата

- преминава се към следващата инструкция

Адресация В инструкциите има 2 адреса за операнди, 1 за резултат, 1 за следващата инструкция и КОП. Общо са максимално 4 но най-често се срещат по 2-3.

T – top of stack

3-адресни инструкции

SUB, ADD, MPY, DIV

2-адресни инструкции

MOVE, SUB, MPY, ADD, DIV

1-адресни инструкции

LOAD, STORE, SUB, ADD, DIV, MPY

По-малко адреси за инструкции по примитивни инструкции по-опростен процесор.Повече адреси повече време за изпълнение по-сложни програми.

load A,M1

load B,M2

SUB A, B

store M3, A

load C, M3

add A, C

store M4, A

Изчислителния процес отнема 14 такта.

Загубата на време, може да бъде намалена чрез прогнозиране на следваща команда за изпълнение.Това може да се постигне като се избере команда непосредствено следваща след командата за преход. Така разгледания конвейер няма голямо ускорение. За увеличаването му се използуват следните два прийоми:

Конвейерът е двустепенен, но в първата степен се изпълняват повече действия по цялостното изпълнение на командата, като извличане, декодиране, определяне адреса на операнда. Така се осигурява приблизително еднакви времена за изпълнението на функциите и за двете степени.

В този случай е прието първата степен на конвейера да се означава като IF-устройство (Instruction Fetch Unit), а втората степен като E -устройство (Execution Unit). Между двете степени трябва да има фиксатор.

Най-простата структура е регистър, който съхранява поредната команда за изпълнение, а връзката е твърда. При запълване на този регистър IF-устройство преустановява работата си.

По-голямо разпространение е получил фиксатор съставен от набор от регистри (регистров файл). Така е възможно всяко от устройствата да работи със своя самостойна скорост.

Необходимо условие е конвейерът да има по-голям брой степени, всяка реализираща различен етап от обработката на командата, напр. извличане, декодиране, определяне адреса на операнда и т.н.

С увеличаване броят степени в конвейера не следва пропорционално увеличение на скоростта на обработка.

Причините за това могат да бъдат постулирани от фактите:

- Във всяка степен на конвейера възникват допълнителни загуби на време, които са в корелационна зависимост с прехвърлянето на данни между фиксаторите и изпълнението на различни подготвителни функции.

- При съвременните процесори се предпочита иновативният подход за увеличаването степените на конвейра,като стремително се увеличава обема на управляващата логика, необходима за отчитането зависимостите при обръщението към паметта и регистрите, а така също и за оптимизация при използването на конвейера.

При изпълнението на някои команди се налага да се въведат задръжки. Това е необходимо за избягването на конфликтни ситуации. Затова командата SUB не започва в следващия такт, а след два такта. Същото може да се посочи и за командата, командата ADD.

Разглеждайки конвейерното изпълнение на програмата при наличие на отделни памети за данни и команди ще забележим няколко разлики:

Вариант 2

1.Могат да стартират едновременно в един и същ такт, защото използват данни записани в две различни памети.

2.Отсъстват задръжки при изпълнението на дадени команди, което предполага по-малко време за изпълнение на програмния фрагмент.

Тактовете стават 9

load A,M1

load B,M2

SUB A, B

store M3, A

load C, M3

add A, C

store M4, A

Може да се осъществи и посредством допълнително увеличаване на производителността , чрез въвеждане на конвейр в изпълнителната степен.

Това води до промени в нея, изразяващи се в появата на задръжки при изпълнение на някои от командите (концепцията на Университета Бъркли) или до вмъкване на нови команди от типа NOP (концепцията на Университета Станфорд).

Тук формално се получава увеличен брой тактове - 11, но не трябва да се забравя, че тяхната продължителност е по-малка, в сравнение с тактовете в предходните примери. Ако се счита, че продължителността на такта е намаляла два пъти, то производителността се увеличава с 260%, т.е. 2,6 пъти в сравнение с не конвейерното изпълнение.

load A,M1

load B,M2

SUB A, B

store M3, A

load C, M3

add A, C

store M4, A

NOP

И в двата случая целта е да се избегнат възникналите междукомандни зависимости, така характерни при работата на конвейера.

За някои операции са нужни i+1, i+2...команди . Те могат да се стартират и да се изпълнят до завършването на i-та команда. Това различие може да създаде трудности, ако не му е било отделено достатъчно внимание на етапа на проектиране, защото операциите, стартирани от i+1, i+2...команди могат да зависят от резултата на i-та команда. Съществуването на тази зависимост създава смущения, които намаляват максималната производителност на конвейера и следва да се избягват, което се прави при проектирането на конвейера или се анализират и отстраняват по време на изпълнение.

Съществуват 3 класа смущения между произволна двойка команди:

- RAW (Read After Write) - четене след запис;

- WAR (Write After Read) - запис след четене;

- WAW (Write After Write) - запис след запис.

извличане на програмния код.

Конвейерът за реални числа, или FPU (Floating Point Unit) е основно преработен, но независимо от това, той запазва съвместимостта с аритметичния копроцесор 80387 и вграденото FPU на 486. Той представлява 8 степенен конвейер с отделни суматор, умножител и делител. FPU удовлетворява спецификациите на стандартите IEEE 754 и по-новия IEEE 854. Той е проектиран така, че на всеки процесорен такт се изпълнява по една операция с плаваща запетая. Ако втората команда е FXCHG (Floating Point eXCHanGe), могат да се изпълняват и две операции за един такт.

Отделните фази на конвейера са:

IF - Извличане;

D1 - Декодиране на командата;

D2 - Формиране на адреса;

EX - Превръщане на външния формат на данните с плаваща запетая във вътрешен и запис на операндите в регистровия файл.

X1 - първа фаза на изпълнението на операциите с плаваща запетая;

X2 - втора фаза на изпълнението на операциите с плаваща запетая;

WF - закръгление и запис на резултата в регистровия файл;

ER - съобщение за грешка и актуализиране думата на състоянието.

Първите пет нива от IF до WB се делят с U конвейера. Правилата за сдвояване не допускат паралелно изпълнение на команди с цели числа и такива с плаваща запетая.

Само командата FXCHG, която разменя местата на два елемента в регистровия стек, може да бъде изпълнена едновременно с друга команда за числа с плаваща запетая.

Чрез тези основни правила се избягва тясното място, което се обуславя от стеково ориентираната регистрова архитектура на устройството с плаваща аритметика. Общо FPU има осем регистъра, които са изградени на принципа на стека. Зареждането на най-горния регистър в стека най-често се извършва чрез командата FXCHG. Ето защо тя е вградена в архитектурата на Pentium по такъв начин, че да може да се изпълнява паралелно с другите команди на аритметиката с плаваща запетая. За нейното паралелно изпълнение са необходими "квази нула такта", а ако това не е възможно - един такт.

От направеното кратко изложение на работата на конвейерите става ясно, че разрешаването на междукомандните зависимости се реализира чрез изпълнението на изключително строги правила, заложени в управлението на конвейерите. В случая не е възможна промяна на реда на изпълнение на командите.

Конвейерната обработка е едно от средствата за повишаване на

производителността на процесора, чрез въвеждане на паралелизъм по

време Например: [1,2,3].Въвеждането й се счита като естествено развитие на фон Нойманова архитектура. Всички съвременни процесори използват конвейерната (pipelined) архитектура в различните и варианти. Това означава, че всяка команда се изпълнява от няколко апаратни блока, обединени в конвейер. Първият процесор, в който се използва това решение е Intel 486, с конвейер от пет фази.

Конвейерната обработка се базира на разделяне на подлежащата за изпълнение базова функция на малки части, наречени подфункции и изпълнение на тези подфункции от отделни апаратни блокове, наречени степени. Подобно на движението на изделие по физически конвейер, командите и данните се преместват по степените на цифровия конвейер със скорост, която не зависи от неговата дължина (броя степени), а само от скоростта, с която новите обекти могат да се подават на входа на конвейера. Тази скорост от своя страна се определя основно от времето, за което един елемент може да бъде обработен от една степен на конвейера (обикновено най-бавната).

Конвейрите от класически тип са 5 - фазови и включват основните процеси при изпълнение на една инструкция:

Извличане на инструкция а Декодиране на инструкция à Изтегляне на операндите à Изпълнение на инструкцията à Запис на резултата.

В идеалния случай при всеки такт на процесора в конвейера влиза една инструкция и излиза резултата от друга инструкция, а тези които вече се намират на някаква фаза се преместват в следващата.

Броят на конвейрите в един процесор може да бъде различен и от тях зависи колко инструкции теоретично могат да се обработят за един такт. Процесор с един конвейер се нарича скаларен. При него в началото на конвейера едновременно може да се намира само една инструкция. По - развитите процесори, които включват няколко конвейера се наричат супер скаларни. Те могат да изпълняват повече от една инструкция за такт.

Ясно е, че с увеличаване на n, S се увеличава и се стреми

към своята пределна стойност L.

От този прост пример се вижда, че конвейерната обработка е

особено подходяща при обработката на дълги вектори.

След направените разсъждения до тук, може да се даде

следното по-строго определение за конвейерна обработка:

Конвейерност означава, че се осигурява съвместяване във времето

на различни действия по изчисляването на базовите функции за

сметка на тяхното разбиване на подфункции. За да се реализира

конвейерна обработка е необходимо да са изпълнени следните пет

условия:

• Изчислението на базова функция е еквивалентно на някаква

последователност от изчисление на подфункции.

• Величините явяващи се входни за дадена подфункция, се

явяват изходни за предходната подфункция.

• Никакви други взаимодействия между подфункциите няма освен

входните и изходните величини.

• Всяка подфункция се реализира чрез апаратни блокове.

• Действията, реализирани от тези апаратни блокове изискват

приблизително едно и също време.

Апаратните средства, необходими за изпълнението на всяка от

тези функции, образуват степен. Всяка степен от своя страна е

изградена от два блока: логически и фиксиращ.

Логическия блок осъществява същинските логически операции,

съответстващи на всяка подфункция, а фиксиращият представлява

свръхбърза памет и служи за подаване на данните в следващата

степен в строго определен момент.

Ако поне едно от посочените условия в определението за

конвейер не е изпълнено, не следва да се говори за конвейерна

обработка. По-правилно е да се говори за препокриване на

операциите. И в двата случая крайният ефект е един и същ -

намалява се времето за обработка, но този ефект се постига с

различни средства.

Под препокриване разбираме:

• когато между отделните обработки може да има някаква зависимост

освен предаването (получаването) на данните;

• за всяка обработка може да е необходима различна верига от

подфункции;

• по своето назначение подфункциите са достатъчно различни;

• времето, необходимо за всяка степен, не е обезателно

постоянно.

За оценка на производителността се изхожда от основното

съотношение - $P\ T$

$S = T1$,

където T1 в случая е брой тактове при последователните

изчисления; Tp е брой тактове при конвейерните изчисления.

В дадената задача времето за изпълнение фрагмента с конвеир и в двата случая е по-бързо от времето за изпълнение програмният фрагмент без конвеир.

Ясно е, че с увеличаване на n, S се увеличава и се стреми

към своята пределна стойност L.

От този прост пример се вижда, че конвейерната обработка е

особено подходяща при обработката на дълги вектори.

След направените разсъждения до тук, може да се даде

следното по-строго определение за конвейерна обработка:

Конвейерност означава, че се осигурява съвместяване във времето

на различни действия по изчисляването на базовите функции за

сметка на тяхното разбиване на подфункции. За да се реализира

конвейерна обработка е необходимо да са изпълнени следните пет

условия:

• Изчислението на базова функция е еквивалентно на някаква

последователност от изчисление на подфункции.

• Величините явяващи се входни за дадена подфункция, се

явяват изходни за предходната подфункция.

• Никакви други взаимодействия между подфункциите няма освен

входните и изходните величини.

• Всяка подфункция се реализира чрез апаратни блокове.

• Действията, реализирани от тези апаратни блокове изискват

приблизително едно и също време.

Апаратните средства, необходими за изпълнението на всяка от

тези функции, образуват степен. Всяка степен от своя страна е

изградена от два блока: логически и фиксиращ.

Логическия блок осъществява същинските логически операции,

съответстващи на всяка подфункция, а фиксиращият представлява

свръхбърза памет и служи за подаване на данните в следващата

степен в строго определен момент.

Ако поне едно от посочените условия в определението за

конвейер не е изпълнено, не следва да се говори за конвейерна

обработка. По-правилно е да се говори за препокриване на

операциите. И в двата случая крайният ефект е един и същ -

намалява се времето за обработка, но този ефект се постига с

различни средства.

Под препокриване разбираме:

• когато между отделните обработки може да има някаква зависимост

освен предаването (получаването) на данните;

• за всяка обработка може да е необходима различна верига от

подфункции;

• по своето назначение подфункциите са достатъчно различни;

• времето, необходимо за всяка степен, не е обезателно

постоянно.

Типичен пример в това отношение е препокриването на входно-

изходните операции с изчислителните – Фиг.4-4.

2. Видове конвейери

Конвейерите биват: еднофункционален и многофункционален.

Еднофункционалният конвейер, както подсказва името му, е способен

да изпълнява един тип базова функция. Многофункционалният

конвейер е способен да изчислява функции от различни типове. В него, в допълнение към входа за данни, има управляващ вход,

регулиращ действието на конвейера.

От своя страна, многофункционалният конвейер може да се

класифицира по честотата, с която се изменя изпълняваната функция на конвейер. Така конвейерът бива със статична конфигурация (статичен конвейер) и с динамична конфигурация (динамичен конвейер).

• Статичен конвейер. - Изменението на типа на изпълняваните

функции са относително редки и те стават непосредствено под

управлението на програмата на програмиста. Пример за такъв

конвейер е аритметичния конвейер във векторния процесор.

• Динамичен конвейер. - Изменението на типа на изпълняваните

функции са чести, непрекъснати (динамични). Този тип конвейер се използува при изпълнение на командите на компютъра. Знаем, че по правило, всяка команда се отличава от предходните по формат и

тип, и следователно е необходимо непрекъснато пренастройване на конвейера по отношение на изпълняваните функции. Понастоящем, в по-голямата част от процесорите се използува динамичен конвейер.

Тука, в отличие от векторните процесори, фактическата реализация на конвейера много рядко се "вижда" от програмиста, а още по- малко се намира под негово управление.

Освен по честотата, с която се изменя изпълняваната функция,

многофункционалните конвейери могат да се класифицират по

пътищата по които данните се придвижват вътре в него. Така те

могат да бъдат еднолинейни (еднопътни) и многолинейни (много

пътни). При еднолинейните, потока от данни при произволна

инициализация върви от една степен към следващата степен,

независимо от типа операция. При конвейерите с множество от

пътища, както подсказва наименованието им, има различни пътища при различните операции.

Проблеми на конвейерната обработка

Увеличената скорост на изчисление води до проблеми непознати

при "класическата" обработка. По-важните от тях са:

Прекъсванията. Системата от прекъсвания, която е достатъчно

добре развита и ясна при класическата архитектура на фон Нойман, при високопроизводителните компютри и в частност при конвейерните търпи сериозно изменение и развитие. Основната трудност идва от това, че няколко фрагмента от операнди или команди едновременно се намират на различни стадии на изпълнение. Както е известно, прекъсванията са асинхронни спрямо нормалното изпълнение на програмата. Така трудността се състои първо да се определи къде трябва да се постави форсиран преход за обработка на прекъсването

и второ как точно да се продължи прекъсната програма след

обработка на прекъсването. Трябва да се подчертае, че с

нарастването на броя степени в конвейера (което означава като

правило и по-висока производителност), се усложняват и

задълбочават посочените по-горе трудности.

Задръжки в конвейера. Задръжките в непрекъснатата работа на

конвейера се определят от структурата или използването на

конвейера и пречат за неговата работа с максимална скорост. Те

биват: структурни и зависещи от данните.

Структурните задръжки възникват когато два различни

фрагмента от данни се опитват да използуват една и съща степен в конвейера едновременно.

По очевидни причини такива случаи се наричат стълкновение.

Задръжките зависещи от данните възникват, когато това, което

протича в една степен от конвейера определя, могат ли данните да преминават през другите степени. Например, две различни степени трябва да използуват общата памет. Когато едната степен използува паметта, то другата, за която също е необходима памет, трябва да"работи" напразно, докато първата не завърши работата си. За разлика от структурните, тези задръжки изцяло зависят от данните и не се поддават на

аналитично изследване.

Между командни зависимости. Този тип проблеми възникват при

конвейерното изпълнение на команди. Едновременното изпълнение на

съседни команди може да влияе на използуваните от тях операнди и

изчислените от тях резултати. Типичен случай на такова влияние е когато i-та команда трябва да прочете някакъв елемент от данни, а i+1-та команда да го измени. С увеличаване на броя степени в конвейера, повече команди се намират в него на различен стадий на обработка и проблемът се задълбочава.

Проблем със запълването на конвейера.

Както стана ясно, за

да работи максимално ефективно конвейера, е необходимо всички степени в него да са натоварени. Ако дължината на обработвания вектор е съизмерима с броя степени на конвейера, или в програмата често се срещат команди за преход, то през по-голямата част от

времето степените в конвейера няма да са натоварени или ще

изпълняват ненужни за конкретния процес команди и от тук следва да се очаква намаляване на производителността на конвейера.