9. Компютърни архитектури. Формати на данните. Вътрешна структура на централен процесор – блокове и конвейрна обработка, инструкции.

Обща структура на компютрите и концептуално изпълнение на инструкциите, запомнена програма. Формати на данните – цели двоични числа, двоично-десетични числа, двоични числа с плаваща запетая, знакови данни и кодови таблици. Централен процесор – регистри, АЛУ, регистри на състоянието и флаговете, блокове за управление, връзка с паметта, дешифрация на инструкциите, преходи.

Обща структура на компютрите.

Под **персонален компютър** разбираме изчислителна машина, използвана от един човек, за решаване на специфични, лични алгоритмични задачи и проблеми. Специфичните особености на персоналния компютър са, че той заема малък обем, има проста структура и система от команди, има ограничен обем на основната памет и опростен интерфейс, към който се свързват всички устройства в изчислителната система.

Въпреки голямото разнообразие на производители и размери, персоналните компютри се характеризират с еднакъв модел на вътрешна архитектура. Този модел е изграден от три основни компоненти:

- централен процесор;
- основна памет;
- входно-изходни устройства (периферни).

Взаимовръзката между основните компоненти определя логическата организация на съответната компютърна система.

Обменът на сигнали, команди и данни се извършва по специални информационни линии. Съществуват два вида логическа организация на компютърните системи:

1. Индивидуални информационни канали – основните компоненти се свързват по схемата на пълен граф, т.е. по схемата “всеки-всеки”;

2. Обща информационна шина - представлява специализирано устройство, отговарящо за пренасянето на информация между компонентите на компютърната система. Схематично организацията е следната:

ЦП = Централен процесор
ОП = Оперативна памет
ВИ = Входно-изходно устройство

Предимството на втората организация е, че тя има ниска цена, проста е, добре е структурирана и е гъвкава, т.е. възможна е лесна надстройка или замяна на основните компоненти. Основен недостатък на втората организация е неизбежното ограничение, че в даден момент могат да комуникират само два компонента, и произтичащата от това по-ниска производителност.

Шината представлява магистрала, по която може да бъде обменяна информация между две или повече устройства.
Шината е споделен ресурс, който всички устройства в конкурентен режим се мъчат да си разпределят. На ЦП се дава пълен приоритет, когато трябва да използва шината. Налага се въвеждане на специално устройство, което се нарича контролер на шината или арбитър, и то извършва разпределянето на шината, т.е. определя кое устройство за колко време да използва шината. Освен за обмен на данни, шината се използва и за предаване на управляваща и адресна информация. Затова условно шината се разделя на три подшини – **подшина за данни**, **подшина за адреси** и **управляваща подшина**. Подшината за данни прехвърля информация между определено място в паметта или входно-изходно устройство и централния процесор. Конкретното място в паметта или входно-изходното устройство се определя от адреса, който се предава по подшината за адреси. Управляващата подшина пренася електрически сигнали, които контролират комуникацията между централния процесор и останалите устройства. По нея се пренасят команди от централния процесор към устройствата и съответно се получават съобщения за статуса на устройствата. Например, управляващата подшина съдържа две линии за четене и за писане, които определят посоката, в която се пренасят данните. Други линии са линията за синхронизация, линии, по които се предават сигнали за прекъсване, и др.
Шината има следните характеристики:

- **ширина на подшината за данни** – количествена мярка за броя битове, които могат да преминават едновременно по подшината за данни;
- **ширина на подшината за адреси** – определя максималния размер на основната памет, например чрез 16-битова адресна подшина могат да се адресират 64KB памет, с 32-битова шина могат да се адресират 4GB памет;
- **скорост** – определя колко пъти шината може да бъде използвана (заемана) в рамките на една секунда, измерва се в MHz.

По-нататък ще разглеждаме само втората организация (обща информационна шина).

Формати на данните – цели двоични числа, двоично-десетични числа, двоични числа с плаваща запетая, знакови данни и кодови таблици.

В паметта данните се представят в двоичен вид. С помощта на n-битово поле могат да се представят точно 2^n различни стойности.

- **Цели двоични числа**

Първият начин за представяне на целите числа е чрез прав код. При него най-старшият бит в n-битовото поле е знаков – 0 означава положително число, 1 означава отрицателно число. Останалите n-1 бита представят абсолютната стойност на числото в двоична бройна система. По този начин при прав код диапазонът от цели числа, който може да се представи с n-битово поле, е $-2^{n-1}+1$ … $2^{n-1}-1$. При това нулата има две представяния – 000…0 и 100…0. Недостатъкът на правия код е, че събирането и изваждането на цели числа се реализират трудно апаратно.

Вторият начин за представяне на целите числа е чрез обратен код. Отново най-старшият бит е знаков. Разликата от правия код е в представянето на отрицателните числа – то се образува чрез инвертиране на двоичното представяне на абсолютната стойност на числото. Диапазонът при n-битово поле отново е $-2^{n-1}+1$ … $2^{n-1}-1$. Нулата отново има две представяния – 00…0 и 11…1. При обратния код събирането и изваждането се реализират

по-ефективно – числата, независимо от знака си, се събират като числа в двоична бройна система, при това, ако има пренос от най-старшия бит навън, то към резултата се добавя 1.

Третият начин за представяне на целите числа е чрез допълнителен код. Отново най-старшият бит е знаков. Разликата от правия код е в представянето на отрицателните числа – то се образува чрез инвертиране на двоичното представяне на абсолютната стойност на числото (както при обратния код) и след това добавяне на 1. Диапазонът при n-битово поле е -2^{n-1} … $2^{n-1}-1$. Нулата вече има единствено представяне – 00…0. При допълнителния код събирането и изваждането се реализират най-ефективно – числата, независимо от знака си, се събират като числа в двоична бройна система, при това преносът от най-старшия бит навън се игнорира.

- **Двоично-десетични числа**

Има още един за представяне на числата – така наречените двоично-десетични числа. При тях всяка десетична цифра се представя с уникална двоична последователност. Предимството е, че се дава възможност за лесно извеждане на числа. Недостатъкът е, че се усложняват аритметичните операции. Една цифра обикновено се представя с 4 бита, които в общия случай представят стойностите/цифрите/символите 0-9. Така 6 комбинации остават неизползвани. Тъй като компютрите съхраняват данни под формата на байтове, има два начина за съхраняване на 4-битовите цифри: всяка цифра се съхранява в отделен байт (непакетирана форма) или в един байт се съхраняват две цифри (пакетирана форма). Едно n-байтово пакетирано двоично-десетично число може да се състои от най-много 2.n-1 цифри, едната е резервирана за знак.

- **Двоични числа с плаваща запетая**

Реалните числа в паметта се представят чрез рационални апроксимации.

Първият подход е рационалните апроксимации да са с фиксирана точка. С други думи, в n-битово поле, първите m бита представят цялата част на числото, а останалите n-m бита представят дробната част на числото – десетичната точка е на фиксирана позиция. Проблемът на този подход е, че не винаги се използват всички битове.

Вторият подход е рационалните апроксимации да са с плаваща точка. Всяко число с плаваща точка има вида $m.2^p$, където m е цяло число, което се нарича мантиса, а p е цяло число, което се нарича порядък. Казваме, че едно число с плаваща точка е в нормализирана форма, ако най-старшият бит на мантисата е 1.

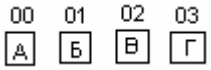
Всяко число с плаваща точка може да бъде нормализирано чрез изместване на мантисата наляво и съответно отразяване в порядъка. Нормализираната форма на едно число с плаваща точка е единствена и обикновено тя се използва за представяне на числото.

Възприет е стандартът едно число да се кодира в 32 бита, като полето за порядъка е 8 бита, а останалите 24 бита са за мантисата и знака. Освен това представяне обаче, за по-голяма точност се използват и двойни думи (по дължина) double precision от 64 бита.

- **Знакови данни и кодови таблици**

За универсално представяне на символите в паметта се постъпва по следния начин: съставя се кодова таблица, в която всеки символ от дадена азбука се асоциира с битова поредица с определена дължина. През 1978г. започва стандартизация на кодовите таблици и днес имаме два стандарта – ANSI и ASCII. ASCII таблицата първоначално е замислена като 7-битова, но впоследствие е разширена до 8-битова за нуждите на различни националности. В различните ASCII таблици символите, кодирани с 0 до 127 са едни и същи, а символите, кодирани със 128 до 255 могат да бъдат различни. Други примери за кодови таблици са EBCDIC – 8-битова кодова таблица на IBM, UNICODE – 16-битова универсална кодова таблица.

Във всяка КС има т. нар. компонент „знаков итератор". Той съдържа памет, която представлява правоъгълни матрици с еднакъв размер, като всяка матрица по битове описва чрез кодиране с 0 и 1 графичното изображение на някой символ.



Първият правоъгълник съответства на двоичен набор 00, вторият – на 01, третият – на 02 и така нататък. Така че, когато трябва да се визуализира графичен символ, програмата издава команда, която се свежда до двоичен код, който визуализира α_1.

ЦП – регистри, АЛУ, регистри на състоянието и флаговете, блокове за управление, връзка с паметта, дешифрация на инструкциите, преходи.

Централният процесор (ЦП) е основен компонент в архитектурата на персоналния компютър. Всяка задача, която се изпълнява от компютъра, директно или индиректно се изпълнява и контролира от ЦП. ЦП осъществява реализирането на машинните команди (инструкции). Някои негови съставни части са:

- управляващо устройство (УУ);
- аритметично-логическо устройство (АЛУ);
- регистри;
- блок за работа с числа с плаваща точка и др.

Регистрите представляват свръхбърза памет, която се използва за съхраняване на информация, върху която се извършват логическите и аритметични операции. Тази информация се съхранява, докато процесорът е под електрическо напрежение или докато някоя машинна команда не я измени. Според функционалността си, регистрите се делят на три вида:

- регистри с общо предназначение (R_1, R_2, …, R_n) – в тях се записват междинни резултати или се използват за адресация; програмистът или компилаторът на съответния език ги използват по свое усмотрение;
- регистри за работа с плаваща аритметика;
- служебни регистри – използват се за специални системни функции, към тях се включват флагов регистър (от голямо значение за управлението и протичането на програмата, съдържа информация за препълване, пренос, посока на обработка на низове, маскиране на прекъсванията и др.), PC (Program Counter, програмен брояч, сочи към следващата за изпълнение инструкция), SP (Stack Pointer, стеков указател, сочи към върха

на програмния стек), MSW (Machine Status Word, съдържа управляващи флагове, служещи за синхронизация на работата на ЦП с ВИ устройства и с ОС); MAR (Memory Address Register, държи адреса на мястото в паметта, където трябва да се изпълни следващата команда); MBR (Memory Buffer Register, действа като буфер, позволявайки ЦП и RAM паметта да работят независимо, без да им влияят малки разлики в работата).

Процесорът обработва стъпка по стъпка последователно, една след друга, командите от основната памет. Регистърът PC сочи в оперативната памет изпълнението на следващата команда. Процесорът изпълнява всички команди вътре в своето тяло и между своите регистри. Всяка команда се извлича от специален регистър в паметта, наречен регистър на командата, и след това процесорът изпълнява командите.

Операциите (аритметически и логически) се изпълняват от специализиран вътрешен блок – **аритметико-логическо устройство** (АЛУ), чиято основна част е двоичният суматор. Извличането на командата се изпълнява от друга част на ЦП, наречена **управляващо устройство** (УУ), което синхронизира работата на вътрешните компоненти на процесора, в това число и на АЛУ, чрез тактови импулси. **Блокът за работа с числа с плаваща точка** също като АЛУ извършва аритметични операции между две числа, но такива с плаваща точка, което е много по-сложно. Сложната структура на този блок включва и няколко АЛУ.

Концептуално изпълнение на инструкциите.

ЦП изпълнява една машинна команда чрез следната последователност от елементарни стъпки, която наричаме **основен/нормален цикъл на изпълнение на командата**:

Първите три стъпки се изпълняват от управляващото устройство (УУ):

1. Адресиране и извличане от основната памет в IR-регистър (Instruction Registry) на предстоящата за изпълнение команда (адресирането става по съдържанието на PC).
2. Обновяване на PC (съдържанието на PC се увеличава с дължината на извлечената команда, тоест се премества на следващата команда, УУ изпраща сигнал към АЛУ за изпълнение на командата).
3. Дешифриране на командата (определят се вида на операцията, броя на операндите, типа на данните и размера на данните).

Следващите стъпки се изпълняват от аритметико-логическото устройство (АЛУ):

4. Локализация на операндите (на базата на информацията от полето на командата по определени методи на адресация се определя местоположението на клетките от паметта, които съдържат операндите).
5. Извличане на операндите във вътрешни MBR регистри.
6. Изпълняване на операцията върху операндите.
7. Записване на резултата (определя се местоположението на резултата и той се занася в съответната клетка на паметта).
8. Проверява се има ли постъпил сигнал от някоя от сигналните линии за аварийна ситуация (за прекъсване). Ако да, управлението се предава на хардуерния авариен механизъм за обработка на прекъсването. След обработване на прекъсването (ако е имало такова) се преминава към стъпка 1 (започва изпълнение на следващата команда).

Типове команди.

1. **Аритметично-логическите команди** се изпълняват върху операнди, които са цели числа със или без знак. Част от тях изчисляват булевите функции конюнкция, дизюнкция, отрицание, сума по модул 2 и т.н. Друга част са операциите за сравнение – равно, различно, по-голямо, по-малко, по-голямо или равно, по-малко или равно и т.н. Последната част са аритметичните операции събиране, изваждане, умножение, целочислено деление и модул.
2. **Инструкциите за преход**. Те променят последователното изпълнение на програмата чрез промяна на стойността на програмния брояч PC. Включват инструкции за условен преход, инструкции за безусловен преход, инструкции за извикване на подпрограма и инструкции за връщане от подпрограма.
3. **Управляващи команди**. Това са инструкции, които се изпълняват, когато процесорът е в привилегирован режим – например чете или пише в служебните регистри. Процесорът преминава в привилегирован режим например при обработка на прекъсване и въобще при изпълнение на системна програма, която е част от операционната система.
4. **Транспортни команди**. Те служат за прехвърляне на данни. Основните са команда за прехвърляне на данни от едно място на паметта в друго, команда за прехвърляне на данни от паметта в регистрите и команда за прехвърляне на данни от регистрите в паметта.

Има и други типове команди – например операции върху числа с плаваща точка (събиране, изваждане, умножение, деление, степенуване, коренуване и др.), операции върху низове (прехвърляне, сравнение, конкатенация), мултимедийни операции MMX (специални векторни операции, които имат голямо приложение при обработката на изображения).

Запомнена програма.

Процесорът изпълнява и реализира командите на програмата. Всяка програма в даден момент от времето се намира в даден етап от своето изпълнение, тоест в някакво състояние. От друга страна, изпълнението на програмата представлява процес, който също се намира в някакво състояние. Това състояние се описва чрез понятието векторно състояние на процеса. Векторното състояние на процеса съдържа цялата информация, необходима за възобновяването на един спрян процес от точката на спиране по такъв начин, че все едно никога не е бил спиран. Включва в себе си: състоянието на програмата (изпълнимия код) в момента на спиране, състоянието на областите с данни, съдържанието на управляващите таблици (блокове) с данни на операционната система, съдържанието на регистрите на процеса и състоянието на входно-изходните устройства. По принцип се счита, че по време на изпълнението си програмата не се изменя, поради което няма смисъл нещо константно (неизменимо) да се съхранява. Счита се, че писането на програми, които се видоизменят по време на изпълнението си, е лош стил на програмиране, и въпреки това и днес той е все още честа практика.