Целта на предмета “Компютърните Архитектури” е да ни въведе в конструкцията на компютрите и да разкрие основните принципи на тяхното действие;

първият съвременен компютър се появява през 1945г. – военен проект на САЩ за бърз калкулатор; създава се от голяма група математици и инженери; името му е ENIAC;

предкомпютърните машини на Чарлз Бабедж и Джон Атанасов не се основават на съвременните компютърни принципи;

Джон фон-Нойман е основоположник на съвременните компютърни принципи;

при Бабедж и Атанасов програмата е отделена от данните;

при фон-Нойман програмата е вид данни и тя се запомня с данните в едно общо поле на оперативната памет;

програмата представлява последователност от команди към процесора на компютъра, наречени инструкции; инструкциите са кратки и еднозначни и те определят какво трябва да прави процесора; те указват на процесора каква операция трябва да се извърши и къде се намират операндите, които участват (т.е. посочва се тяхния адрес);

постулат на фон-Нойман: инструкциите се изпълняват от процесора последователно, т.е. в реда на тяхното записване; те се съхраняват в една и съща памет с операндите и в тази памет няма логическа разлика в това дали съдържанието е инструкция или операнд;

постулатът на фон-Нойман е фундаментален принцип при създаване на съвременните компютри; компютърните архитектури, които се основават на този принцип се наричат

фон-Нойманови архитектури; правени са неуспешни опити за създаване на друг вид архитектури;

с развитието на технологиите се преминава от физически модел на фон-Нойман – физическа памет и физически процесор, а именно:

- изтегля се инструкцията;
- изтеглят се данните;
- изпълнява се операцията;
- записва се резултата;
- преминава се към следващата инструкция;

към концептуален модел на фон-Нойман – при него физическата реализация е по-сложна, например няколко инструкции влизат едновременно в процесора; това, обаче, остава скрито за програмиста и той планира програмите, основавайки се на постулата на фон-Нойман; това е изключително важно: например

нека имаме операция x с операнд с адрес b и резултат с адрес a и операция y с операнд с адрес a и резултат с адрес c в следната последователност:

x : a b

y : c a

това означава, че у използва като операнд резултата от x и за да се изпълни у трябва да завърши x;

цялото програмиране се базира на това, че инструкциите се изпълняват последователно и ако поставим инструкция у след инструкция x, това означава че у ще използва резултата от x;

работата на компютъра се определя от програмите и наредбата на инструкциите в тях; правени са опити да се изгради архитектура, при която данните също да имат роля – например, ако някакъв операнд е готов, програмата, използваща този операнд сама да се изпълни; оказва се, обаче, че е непостижимо за човешката мисъл да използва такива архитектури;

процес на изчисление е събитие във времето, когато във фон-Нойманова машина под управление на запомнена програма се извършва изчисление върху конкретни данни; компонентите, които участват в процеса на изчисление са следните:

- физическата среда (компютъра);

- програмата, която представлява набор от инструкции;

- логическата среда, която се създава от конкретните данни;

- предисторията на процеса – често има голямо значение каква програма се е изпълнявала преди това;

идеалният случай е една фиксирана последователност от инструкции с фиксиран набор от данни да поражда един и същи резултат на всички машини, на които се изпълнява;

задължителният случай е на един и същ компютър една програма при един и същи входни данни да връща един и същи резултат при всяко нейно изпълнение;

целта на предмета “Компютърни Архитектури” е да познаваме същността на реалния изчислителен процес; това има голямо значение – може алгоритъмът, който реализираме да е правилно написан, но да има проблем при самото изчисление;

задачата на компютрите е с тях да може да се моделират изчислителни процеси; представянето на данните в компютрите е основен момент;

първите предкомпютърни устройства представят данните в десетична бройна система; това е физически много трудно постижимо – трябват ни изграждащи елементи с десет на брой различни, устойчиви състояния; пример е устройството на Чарлз Бабедж, което има механична постройка; основен елемент е колело, което чрез завъртане може да преминава в десет състояния; преминаването от едно състояние в друго става за няколко секунди;

изчислителният процес се развива като съвкупност от изменения на състоянията на всички елементи, които участват в процеса; така времето за изпълнение на процеса съществено зависи от времето за преминаване от едно състояние в друго; не е важно изчислителният процес да е само коректен, той трябва да се включва в адекватни граници във времето;

за пръв път през 1945 г. са създадени електронните лампи, които могат да се намират в две състояния – 0 (изключено), 1 (включено) и могат достатъчно бързо да преминават от едно състояние в друго; електронните лампи са сравнително обемисти – това е проблем, тъй като за представяне на информация ни трябват между три и четири пъти повече двоични елементи отколкото десетични елементи;

под един бит разбираме количество информация, за която отговаря един електронен елемент;

първият компонент на компютъра е паметта;

първоначално паметта се изгражда като се определя основна структурна единица, наречена машинна дума – определя се нейната ширината в битове; паметта се изгражда като едномерен линеен масив от еднакви машинни думи; думите в линейния масив се номерират с последователни номера, наречени адреси;

номерацията започва от 0 и стига да последния наличен физически адрес;

концептуално, под адресация ще разбираме следното: формира се адрес, на паметта се подава адрес и тя връща съдържанието на машинната дума на този адрес;

паметта се изгражда като устройство, което може да извършва следните две операции:

- четене – подаване на адрес и извеждане съдържанието на съответната машинна дума;

- писане – подаване на адрес и на машинна дума, която се записва на този адрес; старата информация в машинната дума се губи;

това е логическият модел на паметта;

от архитектурна гледна точка паметта се реализира физически като тримерни матрици;

когато се подаде адрес (двоичен) има устройство, което го дешифрира – адресът насочва устройството към онези елементи, които съдържат съответните битове;

времето за дешифрация и времето за прочитане са крайни времена; под време на достъп разбираме времето от подаването на адреса до извличането на данните; времето на достъп нараства с обема на паметта;

вторият компонент на компютъра е процесора; той осъществява процеса на изчисление, по-точно той изпълнява инструкциите;

структурно процесорът работи по следния начин:

- процесорът взима поредната инструкция от паметта, т.е. адресира поредната инструкция;

- след това дешифрира инструкцията – определят се операцията и се адресират операндите;

- самото изчисление се извършва от аритметично логическо устройство (ALU); по технически причини ALU-то има най-много два операнда; на практика операции с много операнди се свеждат до операции с два операнда; след адресацията на ALU-то са подадени двата операнда, то изчислява резултата и го записва на адрес, посочен в инструкцията;

- след записване на резултата се преминава към следващата инструкция;

това е логически модел на фон-Нойманово изчисление;

много важен елемент в процесора е броячът на командите (адрес на инструкциите); това е запомнящ елемент от регистров тип, т.е. в него се пише без адресиране; регистрите са единични машинни думи, които не са част от паметта;

в регистъра на адреса на инструкциите се поддържа адресът на текущата инструкция, която изпълнява процесора;

една фон-Нойманова машина се намира в постоянен цикъл:

- адресиране и извличане на инструкцията – по съдържанието на брояча на командите;
- дешифриране на инструкцията;
- увеличаване на брояча на командите с дължината на инструкцията; (обновяване на брояча на командите)
- адресиране на операндите;
- изпълнение на операцията;
- записване на резултата;
- преминаване към следващата инструкция;

инструкцията за преход е специална инструкция; тя има основен операнд адрес на

инструкция – този адрес се записва в брояча на командите в предпоследната стъпка;

инструкциите за преход могат да породят безкраен цикъл на изпълнение, въпреки линейното ограничение на паметта;

смисълът на изчислителния процес – да се въздейства на входни данни за да се получат изходни данни; това предполага в компютъра да има входно-изходна система с която той да обменя данни с обкръжаващия свят;

технологически, компютрите се развиват много през последните 50 години; през 1949 г. е създаден транзистор (електронен ключ), по-късно се осъществява интеграция на много транзистори върху силиконова пластина; днес на една пластина с размери два на два сантиметра се интегрират над 100000000 двоични елементи (ключове); напредъкът в компютърните технологии основно се дължи на развитието на полупроводниковата електроника;

големият проблем на електронните елементи е, че трябва енергия, при която се отделя голямо количество топлина; днес интегрални схеми, наречени чипове, са основната съставна част на един компютър;

с развитието на компютрите се оказва, че машинната дума не е най-удобната адресуема единица – това е следствие от нейното разширяване; въвежда се друга основна адресуема единица – байт (B); един байт е осем бита;

ширината на машинната дума се определя от ширината на кой да е от входовете или изхода на ALU-то; в момента стандартът е машини с 32-битова дума, т.е. една машинна дума е 4B;

капацитетът на паметта се мери в байтове;

никоя памет концептуално не може да чете част от байт; физически, броят байтове, които се обменят е по-голям и зависи от броя линии върху шината;

в един двоичен адрес най-десните десет бита могат да адресират един килобайт (KB); най-десните двайсет бита могат да адресират един мегабайт (MB); най-десните тридесет бита могат да адресират един гигабайт (GB); под един килобайт разбираме 2^{10} = 1024 B,

под един мегабайт 2^{20} = 1048576 B, под един гигабайт

2^{30} = 1073741824 B;

така например, с три байта адресно поле могат да се адресират

16MB; с четири байта адресно поле могат да се адресират 4GB;

увеличаването на адресното поле в крайна сметка налага увеличаване на ширината на машинната дума;

капацитета на паметта вече се измерва в MB, GB;

също, компютрите използват и външна памет – до нея достъпът е по-бавен, но тя има по-голям капацитет; основно за външна памет се използват магнитните дискове; тяхният капацитет се

измерва в GB;

понякога за удобство ще изобразяваме битовите количества в тетради и ще ги представяме с шестнадесетични цифри;

в ALU-тата има операции за двоична аритметика;

стандартно целите числа се представят по следния начин:

знаков бит, старши бит, ...

знаковият бит е 1 при отрицателните числа и те са представени в допълнителен код;

знаковият бит е 0 при положителните числа и те са представени в прав код;

колкото по-малко е едно положително число, толкова повече нули има в старшите битове;

колкото по-малко (по абсолютна стойност) е едно отрицателно число, толкова повече единици има в старшите битове;

когато се изпълнява операция с ALU-то и се получи резултат, който е по-широк от ширината на машинната дума, тогава се регистрира препълване;

един прост критерий за препълване е следния – преносът от старши бит към знаков и преносът от знаков бит навън да са различни;

препълването означава некоректен резултат и то трябва да се сигнализира

архитекурно;

друг начин за представяне на числата е чрез десетични цифри, кодирани като двоични (с четири бита); операциите върху такива числа могат да се извършват и от двоично ALU, стига да са осигурени операции за преобразуване от и към двоичното представяне от по-горе; съществуват и десетични ALU-та, които извършват директно операциите с числа, кодирани по този начин;

числата с плаваща точка се представят чрез порядък p и мантиса

m; повечето архитектури възприемат основа на числото да е 16;

характеристика на числото е порядъкът със знак; така стойността на числото е $m.16^p$ или $m.16^{-p}$

в зависимост от знака на порядъка; характеристиките са фиксирани във всички архитектури – обикновено са 7-битови (един бит е за знак);

най-често мантисата е три байта; мантисата има знаков бит, който обикновено е осмият бит в байта на характеристиката;

под нормализация на мантисата разбираме долепяне на значещите битове до десетичната точка (и съответното отразяване в порядъка); мантисата винаги се поддържа нормализирана;

обикновено в процесорите за нуждите на аритметиката с плаваща точка има друго ALU – FPU; в него няма логически операции;

кое ALU ще се използва зависи от операцията, която се изпълнява;

в общата структура на компютъра има два компонента – CPU (процесор) и STORAGE (запомняща част); в STORAGE се пазят инструкциите и данните; частта STORAGE се структурира в йерархия;

CPU трябва да има постоянна връзка със STORAGE за да изтегля данни и инструкции и да записва резултата;

в CPU имаме следните компоненти:

- ALU – аритметично логическо устройство за операции с аритметика с фиксирана точка;

- FPU – аритметично устройство за операции с аритметика с плаваща точка;

- DEC – дешифратор на инструкции;

- PC – брояч на командите, където се формира адресът на следващата инструкция;

по-нататък структурата на процесора се определя от формата на инструкциите; по отношение на този формат развитието на компютърните архитектури е преминало през различни етапи:

- инструкции с фиксирана дължина (обикновено 32 бита); важно е, че всички инструкции на дадена архитектура са с една и съща дължина;

- инструкции с променлива дължина – при тях в зависимост от старшите байтове (обикновено от най-старшия) се определя дължината на инструкцията; тази дължина може да бъде различна; например инструкциите при архитектурата на машините VAX имат максимална дължина на инструкцията 321 байта;

предимства и недостатъци на двата подхода:

- при инструкциите с фиксирана дължина се опростява дешифрирането на инструкциите – ускорява се процеса на дешифрация; такъв формат на инструкциите е удобен за векторни и скаларни процесори; векторните процесори представляват масив от много еднакви процесорни блокове, които едновременно изпълняват една и съща операция, но върху различни операнди (например покомпонентно събиране на два вектора); недостатък на инструкциите с фиксирана дължина е разточителството на памет, тъй като не всички инструкции се нуждаят от цялото поле; също, при по-малка дължина на инструкциите има ограничение на възможността за адресация;

- при инструкциите с променлива дължина дешифраторът постепенно обработва инструкцията и разбира нейната дължина в процеса на обработване; например при архитектурата VAX дължината не се познава още от първото поле; недостатък на инструкциите с променлива дължина е усложняването на дешифриращия блок; самото дешифриране се извършва на няколко цикъла – при архитектурата VAX инструкциите се дешифрират средно на четири цикъла;

архитектурата VAX отпадна по икономически причини; понастоящем най-масово се приема повечето инструкции да са с фиксирана дължина 32 бита, но има изключение за определен набор от инструкции, които са дълги повече от 32 бита;

в общия случай една инструкция съдържа следните компоненти:

- поле за код на операция;

- адрес на първия операнд или самия първи операнд;

- адрес на втория операнд или самия втори операнд;

- адрес на резултата;

в този най-общ вид се разчита на бинарни операции;

по отношение на адресите различаваме три типа архитектури – триадресни, двуадресни и едноадресни;

ако в инструкцията присъстват три отделни полета, указващи адресация на три различни елемента имаме триадресна машина;

при наличие на две отделни адресни полета имаме двуадресна машина; класическият случай е адреса на резултата да е адресът на първия операнд; в зависимост от кода на операцията в инструкцията резултатът може да се запише и във втория операнд;

при едноадресните машини в инструкцията присъства само адресът на втория операнд; първият операнд се подразбира – нарича се акумулатор (ACC) и се намира в специален регистър на централния процесор; по този начин не е необходимо неговото задаване в инструкцията; резултатът се записва в акумулатора;

в общия случай при едноадресни машини програмите

не са по-къси; ако отделните инструкции в една програма работят с различни операнди, то трябва преди всяка инструкция акумулатора да се зарежда с първия операнд;

акумулаторът има своето предимство – например при последователно събиране на десет числа той може да натрупва в себе си междинния резултат; по този начин, ако програмата е строго линейна и всяка инструкция използва резултатът от предишната инструкция, то се намалява дължината на програмата и се увеличава нейната бързина; тъй като акумулаторът е вътре в процесора времето за достъп до него е пренебрежимо малко в сравнение с времето за достъп до оперативната памет;

съвременните компютри се изграждат на цифрови схеми; основата е преобръщането на малки изграждащи елементи, наречени тригери, които могат да заемат две устойчиви състояния, съответни на логическите стойности 0 и 1; тригерите се изграждат на базата на транзистори и техните свойства се определят от протичащия през тях електричен ток; преобръщането на тригер става за крайно ненулево време;

при построяване на логически компютър се счита, че тригерите се преобръщат за безкрайно малко време; на практика това не може да е така, тъй като тригерите са съставени от физически елементи; най-често логическата стойност 1 се асоциира с физически висок потенциал – 5V, а логическата стойност 0 се асоциира с

потенциал 0V;

преобръщането на тригерите става за малък интервал от време, който се натрупва; тригерите може да са свързани каскадно и паралелно; при каскадното свързване тригерите се разполагат един след друг и преобръщането на първия тригер в каскадата води до последователно преобръщане на всички останали тригери в каскадата; броят на тригерите в една каскада е променлив; времената за преобръщане се натрупват и се появява време на закъснение или още време за преходния процес – това е времето за което се преобръщат всички тригери в каскадата;

бързодействието на всеки блок в компютъра се определя от времето за преходния процес; конструкторите на компютрите трябва да могат да градят стабилни цифрови схеми, а времето на закъснение се наслагва от различните блокове; налага се синхронизация във времето – появяват се периодични временни интервали, които да указват началото на поредната стъпка на действие, което се извършва на етапи; например изпълнението на инструкция е многоетапно действие, като се засягат различни цифрови подблокове на процесора; синхронизацията се извършва по следния начин – подава се импулсна последователност от специален тактов генератор с

коефициент на запълване 1 – периодичен импулсен сигнал; неговата продължителност се определя от времената на преходните процеси на отделните подблокове в централния процесор; най-дългия преходен процес на най-сложния подблок трябва да се побере в периода на импулса; смисълът е, че в началото на всеки такт всички микрооперации са завършени; така нареченият overcloaking намалява периода на импулса и схемите започват да работят нестабилно – например една микрооперация започва преди да е завършила предишната микрооперация;

периодът се измерва във време:

- в милисекунди (една хилядна от секундата) – mS;
- в микросекунди (една милионна от секундата) - □S;
- в наносекунди (една милиардна от секундата) – nS;
- в пикосекудни (една трилионна от секундата) – pS;

по-нататък няма смисъл от мерни единици за по-малък период, тъй като скоростта на импулса достига скоростта на физичния максимум – скоростта на светлината;

в развитието на компютрите последователно се преминава от периоди в □S до периоди в nS;

друга характеристика, реципрочна на периода е честотата – колко периода има за една секунда; честотата се измерва в херцове (Hz);

- в една секунда има 1 Hz периода;

- в една милисекунда има 1 KHz периода;

- в една микросекунда има 1 MHz периода;

- в една наносекунда има 1 GHz периода;

- в една пикосекунда има 1 THz периода;

в момента централните процесори имат честота в диапазона на гигахерците;

за изпълнението на най-простите инструкции са необходими поне два такта, въпреки че теоретично, по схемата на фон-Нойман, една инструкция трябва да се изпълни на четири такта; това се постига чрез въвеждане на конвейеризация в работата на процесора – така наречения pipelining;

мярка за скоростта на един процесор е неговата тактова честота; трудно може да се очаква физически постигане на повече от десет гигахерцови честоти; оттам нататък повишаване на производителността на процесора може да се постигне само с архитектурни промени;

когато се проектира процесор се определя краен набор от инструкции, пригодени за този процесор;

множеството от инструкции на конкретна архитектура (ISA) определя структурата на компютъра по отношение на програмистите на машинен език; те трябва да разбират тази структура, за да могат да създават коректни програми на този компютър; понастоящем програмистите на машинен език почти единствено създават компилатори за езици от по-високо ниво;

ISA определя функционалността на процесора:

- какви операции поддържа;

- какви са механизмите за запомняне и адресация и как те се използват;

- как се употребява процесора за комуникация с машинни програми и компилатори към машинни програми; в крайна сметка процесорът изпълнява само машинни програми; компилаторите са специални програми за конкретен процесор, които превеждат програми на езици на по-високо ниво към машинни програми;

характеристики на едно добро ISA са следните:

- реализуемост – как ISA ще се реализира в процесора; възможно е с едно и също ISA да се реализират процесори с различно ниво на производителност; всички добри ISA допускат това – например архитектурата INTEL поддържа едно и също ISA за процесори от 4MHz до 4GHz; възможно е да се направи реализация, в която по сложните инструкции се емулират чрез по-прости, т.е. да се изпълняват като последователност от по-прости инструкции; този подход е с ниска производителност и се счита за лош – смисълът е всяка една инструкция да се изпълнява директно от процесора;

- програмируемост – разбираемост на инструкциите; например архитектурата INTEL X86 не е добра от гледна точка на разбираемост на програмите за нея; в началото няма почти никакви изследвания в областта на компилаторите; ASSEMBLER е език, който е директно свързан с машинния език на компютъра – той представлява еднозначно изображение на машинния език върху символи за изображение; до 1975 година масовият случай на програмиране е на асемблерски код; от 1975 до 1985 година повечето програми се компилират, но се счита че добрите програми се пишат на ASSEMBLER; от 1985 година до сега почти всички програми се компилират и се счита, че те са поне толкова добри колкото асемблерските; това се дължи на развитието на теорията на компилацията и на подобрението на някои езици

от по-високо ниво; през последните 20 години архитектурите се развиват в посока, в която най-добър машинен код при компилиране се получава от програми на C; възможно е по-нататък от C да се премине към JAVA;

- съвместимост на ISA във възходящ и низходящ ред в развитието на архитектурата; под възходяща съвместимост разбираме програми за по-стари процесори да могат да се използват в следващи генерации на процесори за тази архитектура; под низходяща съвместимост разбираме програми за по-нови процесори да могат да се изпълняват и на по-стари процесори; низходящата съвместимост е доста по-трудно постижима и практически не се постига сто процента;

от гледна точка на семантичната сложност се получават два крайни лагера ISA:

- инструкционни множества CISC (complicated instruction set computers); при тях ISA има огромно количество елементи; основна характеристика при CISC е наличието на инструкция за преход към подпрограма със запазване на състоянието на извикващата програма и също връщане от подпрограма с възстановяване на състоянието;

- инструкционни множества RISC (reduced instruction set computers); при тях имаме минимално множество от ортогонални инструкции, т.е. действието на една инструкция не може да се опише чрез останалите; стремежът е за бързодействие и минималност по брой тактове за изпълнение – обикновено при RISC всички инструкции за един и същи брой тактове; за RISC-процесори е тежко програмирането на машинен код, тъй като са необходими много инструкции за описание на по-сложни операции;

различни аспекти на ISA:

- формат на инструкциите – определя дължината и начина на кодиране на инструкциите;

- операции – какви операции се осигуряват, тип на данните, формат на

числата, вид на операндите за всяка операция;

различни архитектурни решения се опростяват с наличието на вътрешна процесорна памет (например PC и ACC); тя има голямо приложение при адресацията за двуадресни архитектури; вътрешната процесорна памет се въвежда като блок от регистри – някои общи и други със специално предназначение; общите регистри могат да се разглеждат като акумулатори; в началото регистрите са били 4 или 16 на брой и са се адресирали съответно с 2 или 4 бита; върховото постижение до момента е процесорът ITANIUM на INTEL, който има 128 регистъра, които се адресират със 7 бита; вътрешната памет касае структурата и състава на регистрите; например част от регистрите могат да се използват за цели числа със знак, други само за адреси; може да има регистри за числа с плаваща точка; един или повече регистри се наричат стекови указатели; стекът е специална структура за запомняне, която е удобна за адресиране в инструкциите; двете операции със стек са push – за включване и pop – за изключване; добавянето и изключването става само във върха на стека; стековият указател е специален брояч, който с всяка команда за работа със стека автоматично променя своето значение; когато се влиза в подпрограма, цялото състояние на извикващата програма се вмъква във върха на стека; състоянието на програмата е свързано с всички регистри и още някои структури от регистров тип; когато започне връщането от подпрограмата се възстановява състоянието на извикващата програма, което се намира във върха на стека; в паметта стекът може да се реализира по два начина – или дъното на стека е във висок адрес и върхът се движи към ниски адреси при добавяне или стекът нараства от ниски към високи адреси; първата реализация се използва по-често; когато се реализира стек трябва да има регистър, който указва къде е дъното на стека; стекът е празен, ако върхът и дъното съвпадат; мястото в паметта за стека се използва само от него и той има фиксиран максимален капацитет; физически стекът се намира в оперативната памет, но при него няма адресиране; логически стекът е част от вътрешната памет на процесора, тъй като работата с него се командва от стековия указател;

основна характеристика на оперативната памет е размера на адресното поле, който съвпада с дължината на машинната дума; например 32 бита (4 байта) могат да адресират 4GB памет;

съществуват различни режими на адресация:

- абсолютна адресация – в инструкцията е записан самият адрес, т.е. всичките 4 байта; тази адресация е неудобна, тъй като заема много място в

инструкцията;

- относителна адресация – адресът се формира чрез базов адрес и отместване; например базов адрес може да е съдържанието на PC и тогава адресът е на някакво отместване от края на текущата инструкция; възможно е отместването да става спрямо съдържанието на предварително фиксиран общ (или адресен) регистър, наречен базов регистър; съдържанието на базовия регистър може да се променя само със специална операция; предимството на относителната адресация е в това, че отместването може да е по-късо от целия адрес;

някои архитектури изискват изравняване на границите на данните (data alignment); например това означава двубайтови структури да започват на четен адрес, четирибайтови структури на адрес, кратен на 4 и т.н.;

за да се преодолее линейното ограничение на паметта възниква нужда за нарушаване на естествения ред на изпълнение на инструкциите; това се постига чрез инструкции за условен и безусловен преход, инструкции за преход към подпрограма и връщане от подпрограма, инструкции за влизане в прекъсване и връщане от прекъсване; при RISC-процесорите няма запазване на състоянието при влизане в подпрограма; напоследък се появиха инструкции за предсказване; при тях се извършват действия без да се знае какви данни участват; стремежът е да се намалят броя на инструкциите за условен преход в програмите;

видове операции:

- аритметически и логически върху цели числа със знак – събиране, умножение, конюнкция, дизюнкция; изваждането, делението, отрицанието, сумата по модул 2 могат да се изразят чрез тях;

- аритметически операции върху числа с плаваща точка – събиране, умножение, коренуване;

- десетични операции върху цели числа – десетично събиране с корекции и конвертиране от двоични числа в десетични числа и обратно;

- низови операции – прехвърляне на низове, сравнение на низове;

- мултимедийни операции (MMX) – специални векторни операции, които имат голямо приложение при обработка на изображения; при тях ALU се използва наведнъж за няколко по-малки числа;

- операция за прехвърляне на данни от едно място на паметта в друго (move);

- операция за прехвърляне на данни от паметта в регистрите (load);

- операция за прехвърляне на данни от регистрите в паметта (store);