**Никола Иванов Чолаков** е роден през 1944 г. в гр. София. През 1971 г. завършва СУ „Св. Климент Охридски“, специалност Математика. В УНСС е на работа от 1977 г., доктор по икономика - от 1996 г.

Специализирал е в Московския държавен университет „Ломоносов“ (1978 г.) и в Международната организация по труда – Женева (1980 г.).

Бил е консултант на ООН по демография (1984 и 1988 г.), пълномощен министър в постоянното представителство на Република България в ООН в Ню Йорк (1997 г.) и пълномощен министър в посолството на Република България в САЩ (1998 – 2001 г.).

Основни публикации: „Демографските процеси и населението на България“ (1972 г.), „Прогноза за населението на НРБ по окръзи (градове и села) през периода 1970 – 2000 година“ (1975 г.), „Демографическая ситуация, принципы, проблемы и меры демографической политики НР Болгарии“ (1985 г.), „Population ageing and its social and economic consequences“, Economic Studies No. 3, United Nations, Geneva, 1992, „Методологически и методически проблеми при статистическото изучаване на смъртността“ (1992 г.), „Математически методи и модели в животозастраховането и пенсионното осигуряване (2003 г.), „Статистическата информация за естественото движение на населението и миграцията в България и някои насоки за нейното усъвършенстване“, сб. „Научни трудове“, 2003, № 2, Университетско издателство „Стопанство“, УНСС, Migration Statistics, Population Accounts and Life Tables - the Methodology used in Bulgaria, сп. „Migracijske i etnicke teme“ № 2 - 3, Zagreb, Sept., 2003 и други.

# ПРИНОС В РАЗВИТИЕТО НА ПЕРСОНАЛНИТЕ КОМПЮТРИ: ЛЕГЕНДАРНИЯТ COMPUCORP 450

доц. д-р Никола Чолаков

**Посвещава се на известния български учен, статистик и демограф проф. Никола Т. Наумов (1925 - 1995), основател и ръководител на Лабораторията за демографски изследвания към Университета за национално и световно стопанство.**

В хода на организирането на Лабораторията за демографски изследвания към катедра „Труд и социална защита“ в Университета за национално и световно стопанство през 1977 г. стана пределно ясно, че не би могло да се осъществява сериозна научна работа по анализа на демографските данни без подходяща из-

числителна техника и още по-малко да се изгражда база данни за българското население в разнообразни разрези, позволяващи достигането на резонни хипотези за бъдещото развитие и проиграването на различни прогностични варианти. Ръководителят на лабораторията проф. Никола Т. Наумов съумя да убеди държавните и партийните авторитети по онова време, че само с молив и хартия не може да се получат желаните научноизследователски резултати по такива парливи проблеми, като демографските, и да се очаква сериозна научна подкрепа за държавната политика по населението. Ние, сътрудниците в лабораторията, имахме известен опит с изделията на американската фирма Compucorp още като научни работници в Националния институт по статистика, така че посрещнахме идващите от Пловдивския панаир кашони още на вратата с мощно „ура“. Разпакетирането отне няколко секунди, свързването на конфигурацията още толкова и пред нас се яви с цялата си прелест компютърната система Compucorp 450.

Сега ми е трудно да изброя всички разработки, за които съществен принос има този герой – Compucorp 450. Останал е обаче незабравимият спомен за онези дни и нощи, когато насаме с машината разнищвахме поредния проблем в демографския анализ. С течение на годините в компютъра (всъщност на дискети) се натрупаха доста статистически данни и чак към края на осемдесетте години на миналия век успяхме част от тях да прехвърлим на персонални компютри IBM с помощта на сериен интерфейс RS-232C и с терминална емулация 3270 да осъществим постоянна връзка по коаксиален кабел на доста по-модерните персонални компютри с инсталираната в УЕИЦ IBM 4331. Така дните на Compucorp 450 се оказаха преброени и макар че все още е в (почти) пълна изправност, вече е на заслужена почивка (по-скоро безнадеждно забравен) в складовете на УНСС.

В хода на работата по два проекта с ООН в началото на осемдесетте се получи известно финансиране с оглед обновяването на компютърния парк на лабораторията, считан от специалистите на UNFPA (United Nations Fund for Population Activities) за малко остарял. Тогава звездата на небосклона бе Apple II, а се задаваше и новата генерация персонални компютри IBM PC. Както можеше да се очаква, напредъкът по въпроса бе задържан от проблемите с кирилицата. При Apple II такава няма, а и трудно можеше да се намерят принтери с кирилица, за да стане възможно използването на българския вариант Правец 82, който и без друго бе изключително дефицитна стока, и освен това кирилицата на принтера да бъде в пълно съгласие с кирилицата на компютъра. Конфигурацията на Compucorp 450 имаше това невероятно предимство да включва стандартна пишеща машина IBM Selectric Typewritter модел 731 (или 735 за по-широка хартия) с прочутата въртяща се печатаща глава. Така че въпросът с кирилицата се решаваше съвсем просто – със смяна на главата. Това наклони везните към запазване на системата с добавяне на нови две дискетни устройства, допълнителен принтер LA-34 на Digital Equipment за улесняване на програмирането, допълнителни глави за пишещата машина и малко консумативи. Да не забравяме още и за ембаргото на такъв „чувствителен“ износ от САЩ, докато Compucorp 450 беше вече свалена от производство и минала в по-леките категории. Няколко години по-късно лабораторията бе снабдена с един IBM PC/XT и скоро след това – с един IBM PC/AT, които веднага бяха свързани с УЕИЦ и модерната за времето си машина IBM 4331.

## 1. СИСТЕМАТА COMPUCORP 450 КАТО АПАРАТУРА

Набор от изделията на Compucorp, предхождащи системата 450

Американската фирма Compucorp е разклонение на Wyle Laboratories от California, които през шестдесетте години на миналия век са били сред водещите специалисти и производители в САЩ на интелигентни декодери и контролери, програмируеми калкулатори и подобни. Постепенно отделението за програмируеми калкулатори се обособява в самостоятелна производствена единица (1968 г.) с официално име[1] Computer Design Corporation. Фирмата в общи линии има ограничено пазарно присъствие, като повечето ѝ изделия се пласират от други търговци като Monroe (дъщерна на Rockwell), Deitzgen, Sumlock (в Англия), Nippon Calculating Machines (в Япония) и др. Калкулаторите Compucorp са били винаги на изключително технологично равнище и доста специалисти[2] считат, че в това отношение са изпреварили времето си. Освен програмируемост със сравнително голям за времето си обем памет и разнообразни функционални възможности тези машини можеха да поддържат голямо разнообразие периферни устройства: перфокарти и перфоленти, магнитни карти, касетофони и магнетофони, впоследствие и дискетни устройства, пишещи машини и различни принтери, плотери, видеотерминали и др. По-късните генерации – към края на седемдесетте години, представляваха завършени персонални компютри, които, свързани в локална мрежа, образуваха атрактивно офис оборудване.


Compucorp 324G Scientist

Системата Compucorp 450 фактически бележи края на калкулаторния период на фирмата производител, въпреки че и порано изделията ѝ са имали много елементи на компютрите, наричани макар и скромно desk-top calculators. Преди това като научни сътрудници в Научноизследователския институт по статистика ползвахме предходния модел Compucorp 445, който на външен вид и размери наподобява познатите по онова време в България Елка, но функционално представляваше завършен компютър, имитиращ калкулатор, при това снабден с пишеща машина IBM като принтер и касетофон за съхранение на дан-

[1] В случая абревиатурата CDC няма нищо общо с известната в миналото Control Data Corporation.

[2] The History of Compucorp, http://www.oldcalculatormuseum.com/d-compucorp.html.

ни и програми. На касетофона може да се ползва безконечна лента, което е удобно за съхранение на верига от програми за последователно изпълнение. При ограничения капацитет на оперативната памет това се оказа изключително удобно. От по-ранното производство на Compucorp заслужава да се отбележи портативният програмируем калкулатор Compucorp 324G Scientist (с размери: 14 см x 23 см x 7 см и тегло 1.5 кг), получил широко признание в американските научни среди.

Системите 445 и 450 заедно с 425 Scientist и вариантите за бизнес приложения 402, 481 и 485 и системата 403 Data Collection System са от едно поколение, като 402, 403 и 450 са върховите модели. Моделът 450 е флагманът на серията, предназначен за научни изследвания (по днешната терминология number cruncher). В основата на всички е процесор, съставен от пет LSI схеми[1], което наподобява и може да се разглежда като прототип на първия едночипов микропроцесор Intel 4004. Процесорът се води като Compucorp 3000 Processor, но елементите му са производство на Texas Instruments и AMI Semiconductor по проекти и техническа документация на Compucorp. Самите компютри като функциониращи системи, както и техните елементи се проектират, моделират и изпитват в Compucorp с помощта на най-мощния по онова време суперкомпютър CDC 6600. Процесорът 3000 има скорост един такт за около 80 микросекунди, което го прави доста по-бавен от Intel 4004 и процесора 6502 на Apple II (Правец 82). Изпълнението на процесорните инструкции отнема един или два такта и само една е многотактова (до 16 такта), така че доста се компенсира в бързодействието. Процесорът е от тип little-endian, което означава, че младшите битове и байтове се разполагат на младшите адреси. Привидно осембитов е, но фактически отделните байтове се обработват на две части по четири бита.

По онова време електронната памет е била доста скъпа и калкулаторите с този процесор често са се предлагали на пазара с памет от 4 или 8 вместо с максимума си 16 килобайта. Процесорът Compucorp 3000 Processor обаче приключва мисията си със серията 400 и най-вече с последните модели конфигурации като 450. След това фирмата се ориентира към новия за момента микропроцесор Zilog Z80 и следващите ѝ изделия се основават на него.

Системата 450 представлява голяма (по днешните стандарти) метална кутия, съдържаща цялата електроника, в т.ч. контролерите за дискетните устройства, плюс два телекомуникационни контролера по стандарта на RS-232C. В кутията са монтирани две осеминчови дискетни устройства, като дискетите са с капацитет 640 килобайта. Години по-късно петинчовите дискети с много зор ще достигнат капацитет от 360 килобайта.

[1] Large Scale Integration. Compucorp означава този чипсет като ACL. Първата машина с този процесор се появява през 1972 г.

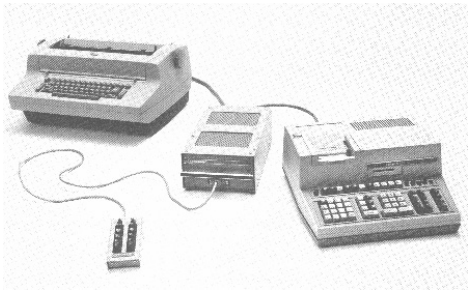
Compucorp 445 Statistician

Предвидена е възможност за включване допълнително на други две дискетни устройства в отделна кутия със собствено захранване, т.нар. Compucorp 491 Disk Memory System. Същата може да се използва и при останалите модели от серията 400 като разширение на конфигурацията. Съществено улеснение за подобно разширение е фактът, че в Compucorp 491 Disk Memory System е предвидено място за инсталиране на телекомуникационните контролери за управление на терминали, нещо, което иначе отсъства в оригиналната конфигурация на моделите 445 и 425. Подобно разширение се използваше например в Централната лаборатория по оптическо запис и обработка на информацията на БАН, с която поддържахме тесни контакти и получавахме ценни напътствия за работата с компютъра, за което сме им безкрайно благодарни.

В системата 450 единият от двата телекомуникационни контролера е ангажиран с клавиатурата и видео дисплея. Клавиатурата на системата е нещо съвсем ново за Compucorp и е ясен сигнал за окончателно скъсване с калкулаторната идеология. Клавиатурата е функционално аналогична на тази от появилото се десетина години по-късно IBM PC и идеално се връзва на CRT дисплея, като заедно образуват основата на удобно работно място.

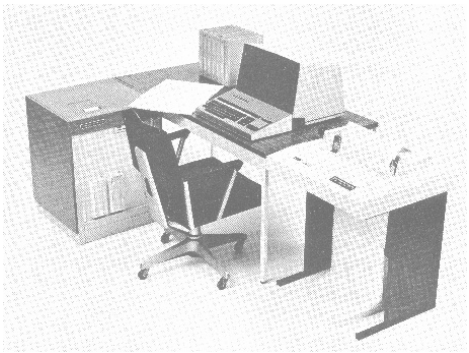
Вторият телекомуникационен контролер използвахме за свързване на бърз иглен принтер – терминал DEC LA-34 за улесняване на програмирането.

Един от „рецидивите“ от по-старите модели е малкият барабанен принтер на горния капак на системната кутия. Изглежда не може да се мине без подобно устройство, което издава сигнали при стартиране на системата или в някои аварийни ситуации, на които останалата периферия не би могла да реагира или просто не е уместно да се смесва потребителска информация със системни съобщения. Принтерът е на Seiko с двадесет и един знака на ред, като за всяка от тези позиции има шестнадесет разновидности на барабана. Най-левият знак обикновено е за валута. Следващите петнадесет знака са за цифри и препинателни знаци като точки, запетайки, тирета и интервали, но и за служебни (по-никога „фатални“) съобщения като ERROR и OVERFLOW. Най-десните пет позиции съдържат специални знаци, и цифри в курсив, за пояснява̀не на резултата. Нормалните резултати се отпечатват с черната част на карбоновата лента, а отрицателните величини (съгласно счетоводните традиции) и някои от аварийните съобщения – с червената. Опитът ни показва, че използването на този принтер за извеждане на резултати от практическата ни работа просто не се налага. Принтерът е, разбира се, напълно програмируем и може да се използва за различни цели, но в нашата практика това не се наложи.



IBM Selectric Typewritter,
свързана с Compucorp 445

Вторият от „рецидивите“ е устройството за четене и писане на магнитни карти. Картите са двустранни по 256 байта, общо 512, и при модела Compucorp 450 службата им се свежда единствено до стартиране на дисковите устройства и съответната операционна система. При по-старите системи от серията Compucorp 400 магнитните карти биха могли да се използват за съхранение на програми, което практикувахме преди като сътрудници в НИИ по статистика, паралелно с касетофона, използван предимно за данните. На лицевата част на устройството има плъзгащ се превключвател, с който се определя дали предстоящата операция е за четене или запис на магнитни карти. Работата с това устройство наподобява банкоматите.



Compucorp 450 в базисна конфигурация

От по-старите модели в системата Compucorp 450 бе запазена най-ценната за нас периферия: прочутата пишеща машина IBM Selectric I/O Typewriter със съответния интерфейс Compucorp 495 Typewriter Controller. Пишещата машина има входно-изходни възможности и може да служи за терминал на всичките модели от серията 400. На снимката се вижда и специализираният контролер Compucorp 495 Typewriter Controller за осъществяване на връзката, към който е прикачена малка подвижна клавиатура за управление на компютъра от работещия на пишещата машина.

С помощта на печатаща глава с кирилица фактически нямахме проблеми с азбуките. Всички текстове на български, а най-често това са заглавията, анотациите на таблиците и други пояснителни текстове, се въвеждат откъм пишещата машина и влизат в компютъра като седем битови ASCII кодове[1]. Поради това никакви проблеми нямаше дори и ако кирилицата се въвежда в изходния текст на програмите.

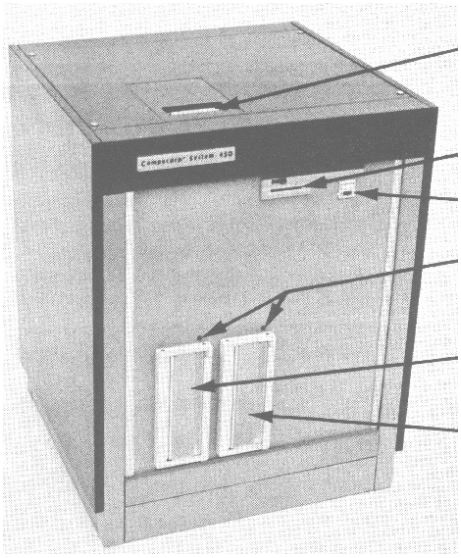
За съжаление системата не притежаваше софтуерни средства за текстообработка и единствената възможност за корекции беше текстовете да се изобразят на екрана чрез латиница с оглед на това да се коригират пасажите и най-

[1] American Standard Codes for Information Interchange. Световен стандарт за кодиране на информация, възприет от началото на шестдесетте години на миналия век. Мнозинството от периферните устройства интерпретират данните от и към компютъра по този стандарт.

вече преформатират страниците и накрая да се отпечатат обратно на пишещата машина в нормален за четене вид. Съвсем ясно е, че подобна примитивна процедура е доста досадна работа, но фактически спестяваше доста машинописен труд.

Друг проблем беше, че CRT дисплеят не беше производство на Compucorp и се оказа невъзможно да се получи документация и да се разбере как да се подмени генераторът на знаците така, че да му се „присади“ кирилица. Тази идея изглеждаше малко безумна, тъй като по всичко личеше, че чипът може да адресира само 128 позиции за стандартната ASCII таблица.

Освен това дисплеят е само текстов, 80 знака на 25 реда и няма никакви графични възможности. По онова време графичните станции се считаха за космическа технология под дебело ембарго и на фантастични цени.



Най-съществената част на Compucorp 450

На снимката в ляво се вижда системната кутия на Compucorp 450. Стрелките отгоре надолу сочат към: (1) системния принтер на горния панел под собствено капаче, (2) на предния панел – устройството за четене и писане на магнитни карти, (3) до него в дясно бутон Power On/Off със светлинен сигнален диод, а не отзад, както беше в началните варианти на IBM PC и човек трябваше да се протяга и да напипва с пръсти ключа, (4) светлинни индикации за работата на двете дискетни устройства, (5) вратичките на дискетните устройства. Металната кутия на системата 450 я прави не само механично устойчива, но и напълно екранира компютъра от околната среда. Ние съхранявахме системата в лабораторията на малка масичка, за да не засмукват охлаждащите вентилатори прах от пода, нещо, което би трябвало да се спазва и при днешните персонални компютри.

На предходната снимка пред стола се виждат CRT дисплеят и клавиатурата, които са свързани със системата с достатъчно дълъг кабел. Вдясно от тях е матричен принтер DEC LA-36 на специална стойка. Вляво от CRT на масата се оформя малко свободно пространство и на него са оставени няколко кутии с дискети във вертикално положение, както се препоръчва от специалистите.

## 2. СОФТУЕРНО ОСИГУРЯВАНЕ НА COMPUCORP 450

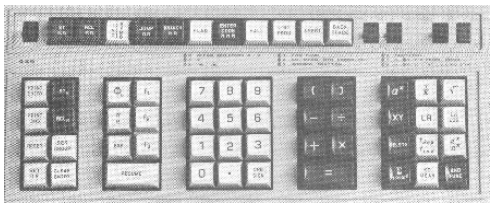
В общи линии системният софтуер във фамилията 400 се намира в ROM (Read Only Memory, по днешната терминология това е BIOS) памет на машината

(около 7 килобайта). Външен софтуер с развойно или приложно предназначение практически няма. Системата 450 прави обаче едно съществено изключение със своята дискова операционна система и асемблер. За езици от високо ниво и дума не може да става. В началото на седемдесетте години на миналия век за BASIC е твърде рано.

Срещали сме някои приложни програми в областта на статистиката, използващи клавиатурата за въвеждане на данни и системния принтер – за показване на резултатите, което ги прави съвършено непригодни за научноизследователска работа. Освен това тези програми страдаха от редица конструктивни ограничения, а и бяха съвсем „учебникарски“, така че всеки грамотен човек би могъл сам да си направи подобен (че и по-добър) софтуер. Подозирам, че създаването на такива програми е било само с образователна цел.

За да се разбере проблемът със софтуерното осигуряване на Compucorp 450, трябва да се има предвид, че цялата фамилия 400 носи генетичния белег на калкулаторите. Така например програмирането на Compucorp 425 и 445 без дискова памет изглеждаше по следния начин. С един от ключовете на клавиатурата компютърът се превключва от RUN на LOAD. От клавиатурата се набират поредните команди и накрая се превключва обратно на RUN. Така въведената програма остава в паметта на компютъра до неговото изключване и е уместно все пак да се запише на магнитни карти или лента. Може програмата да се разпечата и да се инспектира последователността от така въведените макрокоманди. С бутона RESUME може програмата да се стартира и да се проследи действието ѝ. На практика това означава, че компютърът запомня последователността от работата на човека, след което е в състояния да я възпроизвежда многократно без външна намеса, освен когато трябва да се въвеждат данни. Това в общи линии е философията на програмируемите калкулатори.

Наборът от макрокоманди е силно развит, в т.ч. условни и безусловни преходи към различни места от програмата и подпрограми – както с директно, така и с индиректно адресиране, и разнообразни манипулации с паметта за данни, като индиректното адресиране позволява операции с масиви от данни. Съществува и символично адресиране, като всеки бутон от клавиатурата, освен за съответния макрос може да служи и като символичен адрес както за програмите, така и за данните. Налице са всички аритметични действия, множество математически функции и основните статистически процедури. Голяма част от макрокомандите не е изведена на клавиатурата със съответни бутони и за целта е предвидено ръчно набиране на съответния код с помощта на специален бутон ENTER CODE, последван от трите цифри на кода на макроса. Например Compucorp 425 има на клавиатурата си доста математически функции, но липсват статистическите. При Compucorp 445 пък е обратното.



Клавиатура на моделите Compucorp 425 и 445, непосредствено предхождащи системата 450

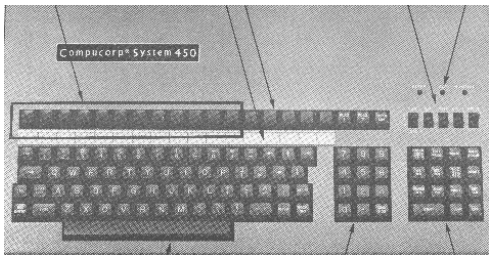
Осигурена е гъвкава възможност за трасиране и редактиране на програмите. На клавиатурата има специализирани бутони за премахване или вмъкване на инструкции. Превключвателят RUN / LOAD има междинно положение STEP за трасиране на програмите. При трасиране се отпечатват на системния принтер последователните стъпки на програмите и по желание на програмиста може да се отпечатват и резултатите от действието.

Съществено място сред макросите заемат тези за входно/изходните операции и особено за управлението на магнитните карти и ленти и на принтерите. За щастие тези операции са съвсем унифицирани и с тях се борави лесно. Съответният макрос се състои от поредица байтове, означаващи за кое устройство се отнасят, коя е операцията, какъв е обемът на данните и от кой адрес нататък се намират или трябва да се разположат в паметта на компютъра. Това е абсолютно същата поредица, ако с машината се работи като с калкулатор, а не в програмиращ режим LOAD. И в този дял идеологията на програмируем калкулатор е спазена стопроцентово.

Изработването на програми за системите Compucorp 425 и 445 все пак се оказа доста тромаво. Листингите от системния принтер представляваха досадни ленти от хартия, които отгоре на всичко се нуждаеха от допълнителен коментар, добавен на ръка. Ако сравняваме това с програмистката практика на големите машини като популярните по онова време ЕС 1020, ЕС 1030 и подобни, предпочитанията все пак остават към Compucorp. Няма чакане за перфорация и за машинно време, не се накъсва целият процес на изготвяне на програмите. Трасирането, експериментирането и апробирането на програмите се вписват също лесно в процеса. Самостоятелността и независимостта от странични и външни фактори позволяват модулно програмиране с изготвяне на ключовите възли от програмите и постепенното им надстройване с допълнителните елементи. В крайна сметка с машини като Compucorp се пестят време и нерви, по-лесно и по-бързо се отстраняват дефекти и програмните продукти се оказват по-надеждни. Има, разбира се, и недостатъци: тези програми не могат да се използват на компютри, различни от тази фамилия на Compucorp, няма мобилност, никаква портативност на софтуера. За работата в лабораторията обаче подобни минуси едва ли са били съществени, тъй като изготвяните програми бяха само за вътрешна консумация.

Системата Compucorp 450 осъществява решителен поврат в практиката на програмирането на машините от серията 400, като фактически скъсва с калкулаторната рамка на предходните модели и с основание тази конфигурация може да се нарече компютър, при това персонален компютър. Подобно усъвършенстване би могло да се постигне с добавяне към системите Compucorp 425 и 445

на дискови устройства, които съдържат и необходимите телекомуникационни контролери за поддръжка на видео или принтер терминал.




Клавиатура на Compucorp 450

Съществен момент в системата Compucorp 450 е наличието на асемблер, т.нар. AL400, представляващ компилатор с редактор за въвеждане и коригиране на програмния текст. С помощта на това средство изготвянето на програмите неимоверно се улеснява и прави клавиатурите на моделите 425 и 445 напълно излишни. Поради това при модела 450 такава клавиатура липсва, а вместо нея има стандартна буквено-цифрова клавиатура, аналогична на тази при сегашните персонални компютри. Само в дясната част на клавиатурата се намират няколко бутона за макроси – главно за въвеждане на цифри, както и превключвателят RUN / LOAD / STEP, което явно служи за аварийни ситуации. На последната фигура е показана такава клавиатура. Стрелките отгоре сочат последователно (от ляво на дясно) към (1) специализираните бутони, означавани на съвременните клавиатури като F1, F2 и т.н. до F12, (2) хартиената лента за маркиране на тези бутони, (3) последните осем бутона, имащи по-специални функции, (4) пет превключвателя, вкл. RUN / STEP / LOAD, и (5) три светлинни индикатора. Долните стрелки сочат към (6) стандартната част от клавиатурата, (7) цифровите бутони и (8) някои бутони за директно управление на компютъра. Последните две групи са в общи линии наследство от предходните модели от серията 400 и използването им при модела 450 се налага само в извънредни ситуации.

## 3. ДИСКОВА ОПЕРАЦИОННА СИСТЕМА

Дисковата операционна система улеснява допълнително развойната дейност, като дава възможност за много по-прегледно разполагане във файлова система на програмите и данните за тях и значително улеснен и по-бърз достъп до необходимата информация. Всъщност преди появата на DOS е използвана друга система DDD (Direct Disk Driver), която представлява адаптация за дискети на софтуера с последователен достъп за управление на магнитни ленти, поради което не предизвиква особен интерес. От противоположната страна на идеологията съществува FBD (Fixed Block Driver), аналогична на известната програма RWTS за Apple II (Правец 82) или INT 13h на IBM PC, за директно писане на писти и сектори по дискетите. В ЦЛОЗОИ използваха предимно FBD поради високата скорост на работа със сравнително обемисти масиви, но с проста структура. В работата на лабораторията ни се налагаше да боравим с файлове, чиито записи имаха доста сложен йерархичен строеж, което ни насочи към DOS.

По функционалност DOS е доста различна от DDD и FBD. Тя съществува в три варианта: DOSN, DOSK и DOSH, като най-простият е DOSN. При него

има директен достъп до отделните записи във файловете по техния номер, нещо като системата DDD, но доста по-култивирано. Възможен е, разбира се, последователен достъп запис по запис напред и назад като при магнетофонните ленти. Вариантът DOSN също не представлява особен интерес. Той е фактически DDD с разширение за директен достъп.

Вариантите DOSK и DOSH са доста по-интересни и много по-полезни. Те включват в себе си DOSN, като освен това търсенето и сортирането може да стават по съдържанието на отделните записи, не само по номера им. При DOSK за всеки файл може да се създаде един или няколко т.нар. асоциирани файла в зависимост от критериите за сортиране на съдържанието на основния файл. Асоциираните файлове съдържат само логическия ред на ключовите сортировъчни полета на фактическите записи, имат много по-малък обем и са винаги сортирани, в резултат на което и търсенето в тях става неимоверно по-бързо (например чрез деление на две части[1]). От асоциирания файл се извлича физическият номер на фактическия запис в основния файл и така се осъществява намирането на желаната информация. Последното е напълно автоматично и невидимо за работещия (човек или програма) на компютъра, все едно, че основният файл е сортиран.

Системата DOSH е изтънчена и представлява забележително технологично постижение за времето си. Тя съдържа DOSK и освен това дава възможност за търсене и намиране на информацията по метода hashing[2]. Хашираните файлове не се нуждаят от допълнителни метаинформационни асоциирани „спътници“ и намирането на търсената информация става чрез кодиране на ключовите полета и трансформирането им в числови величини, определящи сортировката на фактическите данни. За съжаление може да се случи ключовите полета на два различни записа да резултират в еднакви числови кодове, при което би настъпил колапс в системата. Поради това в хеширането се предвижда резервен механизъм като свързан списък (linked list), в който временно се съхраняват квазидублиращите се записи до отстраняването на противоречието. Оказва се, че колкото повече празни записи има един файл, толкова по-рядко се случва подобен конфликт. Нашият опит с хашфайловете показва, че времето за достъп до определен запис от такива файлове е от порядъка на това при обикновения директен достъп и се уверихме, че технологията на хеширане е едно значително постижение в информатиката.

Ако направим сравнение на Compucorp DOS с дисковата операционна система на Apple II и IBM PC (Microsoft DOS)[3], не може да не направи впечатление, че Compucorp DOS по много показатели е ненадминат и днес. Това, което му липсваше, бяха някои служебни функции – примерно за копиране на файлове. Тук е мястото да се отбележи обаче, че в Compucorp не споделят идеята за копиране на файлове, както това се наложи по-късно при Apple II и IBM PC. В Compucorp

[1] Вж. Donald Knuth: *The Art of Computer Programming, Vol. 3: Sorting and Searching*, Second Edition, Reading, Massachusetts: Addison-Wesley, 1998, както и Niklaus Wirth: *Algorithms + Data Structures = Programs*, Prentice-Hall, 1986.

[2] Пак там.

[3] А защо не и с UNIX и WINDOWS.

DOS няколко дискети (от една до четири дискети) може да се обединят в общ том (VOLUME) и да няма особено значение къде точно се намира физическото местоположение на файловете. Всъщност за дисковата операционна система определеност има само томът, а колко дискети се съдържат в него, е без значение. Според авторите на Compucorp DOS техническият носител на файловете не трябва да има никакъв смисъл, а ползването на данните да става само с пренасочване на съответните ориентири и указатели, така че копирането да стане съвсем излишно. Според тях прехвърляне на файлове от един том в друг (още повече от една дискета на друга) чрез копиране е сериозна грешка в дизайна, която не бива да се допуска. Те считат, че една дискова операционна система трябва да бъде система за управление на бази данни, а не просто инструмент за боравене с файлове, дискети и дискови устройства. Неслучайно те наричат дисковата си операционна система (в последния си вариант от 1976 г.) DATA BASE MANAGEMENT SYSTEM. Тази позиция в известна степен е дискусионна, но заслужава определено внимание. Тя подсеща за тезата[1] на Дайкстра за паразитността на оператора GO TO в езиците за програмиране. В днешното време на световна информационна мрежа се съдържат доста индикации за това, че специалистите от COMPUCORP са били прозорливи.

Това не се отнася до копиране на дискети обаче. Напротив, от Compucorp съветват ценна информация да се съхранява в няколко екземпляра, копирана на съответни дискети. Това всъщност означава томове с ценна информация да се съхранява в няколко екземпляра и по такъв начин да се „документира“ на дискети целият процес на изграждане на потребителското приложение. Дисковата операционна система не притежава собствен инструмент за копиране на томове и дискети, а такава функция има на стартовата магнитна карта, с чиято помощ се стартира самият DOS след включване захранването на компютъра. Там има още сервизна функция за форматиране на дискети и дори за копиране на отделни писти, а не цели дискети. Възможно е също да се създават стартови магнитни карти, т.нар. DOS LOAD & GO карти, зареждащи само оперативната част на DOS и стартиращи конкретна програма. Системата DOS обаче съдържа в себе си образ на магнитната карта, който може да се стартира без физическо използване на четеца на магнитни карти.

В Compucorp DOS не се предвиждат сервизни програми за сортиране на записите във файловете. Съгласно разбиранията на авторите това се постига с механизмите на системата за управление на бази данни. Настойчиво се окуражава създаването на асоциирани файлове, в случай че се налага друга сортировка при ползването на базата данни. Подобна техника е създаването на паралелни файлове с механизъм на асоцииране, експлицитно изграден от програмиста. Паралелните файлове обаче може да имат много по-малко на брой записи в сравнение с базата данни и така да се постигнат значителна икономичност и фокусираност върху решаваната задача. Самите бази данни се разглеждат като хаотично натрупана информация, като с помощта на системата за управление се въвежда желаният порядък, без да се преработват първоначалните масиви и без

[1] Dijkstra, Edsger W.: *Go To Statement Considered Harmful*, Communications of the ACM, Vol. 11, No. 3, March 1968, p. 147 - 148.

да се губи или дублира информация. По тази причина не се предвижда копиране на файлове.

Физическото разпределение и предназначение на пистите и секторите на дискетите за Compucorp DOS също са особени. От всичките 640 килобайта на дискета 512 са заделени за потребителски файлове с данни и 128 – за програми. Файловете може да се разпростират на няколко дискети (в multi-surface volume), но винаги се спазва това разпределение. Разполагайки с четири дискетни устройства в лабораторията, можехме да си позволим да боравим on line с масиви от данни и съответни файлове с доста по-голям размер, което ни спести обичайните хитрувания при работа без хард дискове.

За постигане на известна сигурност при работа с файловете (както тези с потребителски данни, така и програмите) операциите по дефиниране (откриване), отваряне, асоцииране (при DOSK), преименуване и т.н. на файлове изискват две пароли за четене и писане. Тези пароли се използват и при боравене с отделни записи от файловете при DOSK и DOSH. Пароли има и за програмите, с които се контролират стартирането им, записът на диск, изтриването, преименуването и т.н.

Операциите и функциите на дисковата операционна система изискват набор от параметри (аргументи на функциите), които трябва да са разположени последователно в паметта и напълно определени, преди да се стартира съответната операция. Отделните параметри са осембайтови, като броят на параметрите е специфичен за всяка операция на DOS. Някои операции не изискват параметри. Невалидни или недостатъчни параметри се разпознават от DOS и предизвикват съответно съобщение.

Функциите на Compucorp DOS може да се групират по следния начин:

| Операции и функции | Описание |
|---|---|
| | ***Операции с томове*** |
| DEFINE VOLUME | Създаване на нов том, състоящ се от една до четири празни, но форматирани дискети, или за промяна на броя на дискетите във вече съществуващ том. |
| RENAME VOLUME | Смяна на името на съществуващ том. |
| | ***Операции с файлове*** |
| DEFINE FILE | Дефинира нов файл заедно с паролите за достъп и резервира съответното пространство в тома. В хода на дефиницията се указва колко записа има във файла и колко байта – в отделния запис. Файловете може да имат от един до 65,535 записа. Всички записи в отделния файл са с еднаква дължина от 4 до 4,088 байта. След дефинирането на нов файл всичките му записи са празни (null records).<br>Дефиницията може да съдържа указание за ключови полета, с което файлът става K-файл или H-файл. При липса на ключове файлът остава от най-простия вид N-файл.<br>Всеки запис има т.нар физически и логически номер. Физическият номер се определя от реда на запълването на записите с информация. Логическият номер се определя от сортировката на ключовите полета. При K-файловете DOS създава т.нар директория, в |

| | |
|---|---|
| | която с по два байта се определя логическият номер на всеки запис. При тези файлове не е възможно за два записа да съвпадат всички ключове. Останалата информация извън ключовете може да съвпада.<br>При H-файловете логическият номер се пресмята въз основа на съдържанието на ключовете, което е същността на технологията hashing. А при N-файловете логическият и физическият номер просто съвпадат. |
| LINK FILE | Дефинира асоцииран файл, свързан с друг вече съществуващ файл с потребителски данни (база данни). Асоциираните файлове се състоят фактически само от директории, но формирани по други ключове, различни от тези в базата данни. Базата данни може дори да няма никакви ключове и да представлява N-файл. Максимум седем асоциирани файла може да се свържат с една база данни.<br>Техниката на асоциираните файлове представлява основен инструмент за една или друга сортировка на базата данни, без фактически да се сортират записите в нея. |
| OPEN FILE | Отваря съществуващ файл за установяване на достъп до неговите записи. Дефинирането на асоциирани файлове изисква базата данни да бъде отворена.<br>Във всеки отделен момент един от записите на отворен файл е текущ. Непосредствено след отварянето на N- или K-файл текущият запис е този с логически номер нула. При H-файловете началният запис може да бъде празен (null record). |
| CLOSE FILE | Затваря файл и актуализира служебната информация за него. |
| CLOSE ALL FILES | Много удобна операция при приключване на работата (обикновено далече след полунощ). |
| RENAME FILE | Смяна на името на файла или само на паролите за достъп. |
| ENLARGE FILE | Добавяне на нови празни записи във вече съществуващ файл. Не съществува обратната операция за съкращаване на файлове. Това може да се постигне само с преписване на записите на файла в нов файл с по-малки размери. |
| CLEAR FILE | Превръща всички записи във файла в празни (null records). |
| PURGE FILE | Премахва целия файл. |
| | ***Операции със записи*** |
| READ | Копира текущия запис от даден отворен файл в паметта на компютъра. |
| WRITE | Копира участък от паметта на компютъра в даден отворен файл върху текущия запис. Тази операция се използва само при N-файловете. |
| DELETE | Изтрива се текущият запис, което го превръща в празен (null record). Изтриването на запис в база данни трябва да се предхожда от изтриване на съответните записи във всички асоциирани файлове, за да не се загуби синхронизацията. |
| FIND | Системата се настройва да намери запис по неговия логически номер. Самото позициониране на четящата глава върху новия запис се извършва в началото на последващия READ, WRITE или DELETE. Друг резултат от тази функция е, че във входния регистър на компютъра се получава физическият номер на търсения запис. |

| | |
|---|---|
| BUFFERED FIND | Също като FIND, но с авансово позициониране (фактически off-line) на четящата глава на съответната писта на дискетата. Много ефикасна операция, тъй като по време на нейното изпълнение процесорът е свободен и може да се използва за други цели. |
| FORWARD | Следващият (по логическия номер) запис на файла става текущ. При H-файловете се прескачат празните записи. |
| BACKWARD | Предходният (по логическия номер) запис на файла става текущ. При H-файловете се прескачат празните записи. |
| CRITERIA | Дефинират се релации за набор от ключове. Това служи за друга операция SCAN, с която се намират само онези записи, отговарящи на тези релации. Установените с CRITERIA релации не са свързани с конкретен файл и може да се използват многократно при различни файлове. |
| SCAN | Намира първия запис, отговарящ на дефинираните в CRITERIA условия. Файлът може да е N-файл без никакви ключове или K- и H-файл с ключове, съвпадащи или различни от описаните в CRITERIA. Scan има допълнителен параметър, с който може да се укаже колко записа да се сканират. |
| KEYED ADD | Добавя нов запис в отворен K-файл с данни от паметта на компютъра. |
| KEYED REPLACE | Намира и заменя съдържанието на запис в отворен K-файл с това от определен участък от паметта на компютъра. |
| KEYED READ | Намира и копира в паметта на компютъра запис в отворен K-файл със съответно съдържание на ключовете. |
| KEYED DELETE | Намира запис в отворен K-файл със съответно съдържание на ключовете и го изтрива. |
| HASHED FIND | Системата търси запис в отворен H-файл по зададено в паметта на компютъра съдържание на ключовете.Тази операция е винаги буферирана като BUFFERED FIND. |
| HASHED ADD | Добавя нов запис в отворен H-файл с данни от паметта на компютъра. |
| HASHED REPLACE | Намира и заменя съдържанието на запис в отворен H-файл с това от определен участък от паметта на компютъра. |
| HASHED READ | Намира и копира в паметта на компютъра запис в отворен H-файл със съответно съдържание на ключовете. В редица случаи по-ефикасно може да се окаже използването на HASHED FIND, последвано от обикновен READ. |
| HASHED DELETE | Намира запис в отворен H-файл със съответно съдържание на ключовете и го изтрива. |

***Операции с програми***

| | |
|---|---|
| WRITE PROGRAM | Записва в програмната библиотека на даден том програма от паметта на компютъра. |
| READ PROGRAM | Копира в паметта на компютъра програма от програмната библиотека на даден том. |
| CALL PROGRAM | READ PROGRAM плюс стартиране на програмата. |
| DELETE PROGRAM | Премахва програма от програмната библиотека на даден том. |

***Системни функции***

| | |
|---|---|
| INITAILIZE и SILENT INITAILIZE | Проверява състоянието на дисковата операционна система, дискетните устройства и томовете в тях и подготвя сведение за констатираното положение. Същевременно се правят опити да се |

| | |
|---|---|
| | коригират намерените дефекти в състоянието на системата. При SILENT INITAILIZE извеждането на съобщения на системния принтер е съвсем ограничено до фаталните. Тази операция се среща при работа с много дискети, когато в хода на програмите се налага те да се сменят. |
| LIST LIBRARY | Извежда на терминала подробен списък на програмите от програмната библиотека на даден том. |
| PRINT MESSAGE | Отпечатва съобщение от DOS. В много случаи е практично използването на функциите на DOS да се настрои така, че системният принтер да не реагира на грешки. В такъв случай PRINT MESSAGE може да се използва при неочаквани и извънредни ситуации. |
| LIST FILES | Извежда на терминала подробен списък на файловете от даден том. |
| GENERATE DOSK | Създава системна дискета с Compucorp DOS, т.е. том с Compucorp DOS като член на програмната библиотека. Такива томове се наричат MASTER VOLUME. Не е задължително томовете да имат DOS. |
| MONITOR | Стартиране на диалоговата система на DOS. |

Операциите и функциите на DOS имат идентификационен код. Например кодът на OPEN FILE е 002 в осмичен запис, на KEYED READ – 035. Съответно на това имат символични имена в асемблера. Символичното име на 002 е OPNF, а на 035 – KRED.

В Compucorp DOS има надстройка – т.нар. монитор, за диалогов режим с потребителя и ръчно управление на функциите на системата. Това е доста аналогично на познатия команден промпт в Microsoft DOS (и Windows), осъществяван от програмата command.com (и cmd.exe в по-новите версии на Windows). Мониторът е удобен за стартиране на програми, преглеждане на каталозите на файловете и програмите, за дефиниране на томове и файлове, експериментиране с функциите на DOS и въобще за операции, за които обикновено не се налага някаква автоматизация. В монитора се намира командата BOOT, с която се симулира прочитане на стартовата карта от устройството за магнитни карти.

Естественият завършек на действието на приложените програми е със стартиране на монитора. Това се познава по появата на съответния промпт (...) на терминала.

## 4. СОФТУЕРЕН МОДЕЛ И ФУНКЦИОНАЛНА СТРУКТУРА НА COMPUCORP 450

Моделът Compucorp 450 е с 16 килобайта оперативна памет, това е максимумът на възможностите на процесора да адресира. Няма да се спираме на някои от по-старите варианти от серията 400 с 8 или дори 4 килобайта, особено компютрите с образователно предназначение. Паметта на Compucorp 450 е разделена на четири страници от по 4 килобайта. Първите осем килобайта са RAM (Random Access Memory) за съхранение на системни програми и данни като

Compucorp DOS или асемблера AL400, както и за потребителски макива. Следващите седем и половина килобайта са ROM със системен софтуер, осигуряващ в крайна сметка правилното функциониране на макрокомандите и необходимия уют за потребителите. Последните 512 байта са RAM, който се използва от системния софтуер като работна област за маневриране и съхранение на междинните резултати при изпълнението на макросите. Процесорът има специализирани високоскоростни инструкции за боравене с този RAM.

Страниците се считат номерирани от 0 до 3. Всяка страница съдържа 4,096 байта, разглеждани на 16 групи (т.нар. колони) от по 256 байта. В осмичен код адресите на колоните стават:

| Страница | Колони | Страница | Колони |
|---|---|---|---|
| 0 | 00 – 17 | 2 | 40 – 57 |
| 1 | 20 – 37 | 3 | 60 – 77 |

Байтовете в отделните колони се адресират с местоположението им спрямо началото на колоната. Първият байт има адрес 000, а последният – 377. Заедно с адреса на колоната байтовете придобиват уникален адрес в цялата памет на компютъра. Така началният байт от цялата памет има адрес 00 000, а последният – 77 377. Например на адрес 47 163 се намира началото на програмата за пресмятане на квадратен корен. Това, разбира се, е в ROM, по-точно в страница 2. Последните 512 байта от страница 3, представляващи служебния RAM, заемат колоните 76 и 77. В COMPUCORP е възприето осмичният запис на числата с размер един байт да е с три осмични цифри. Най-младшата е за най-младшите три бита. Следващата е за следващите три бита и най-старшата е за най-старшите два бита. Така осмичният запис на числата от 0 до 255 става от 000 до 377. Например числото 185 в осмичен запис става 271, тъй като по битове (в двоичен запис) изглежда така: 10-111-001.

Друг важен термин при Compucorp е регистър. Под това се разбира поредица от осем съседни байта (64 съседни бита) от паметта, като началният (най-младшият от тях) има адрес от вида YY XX0. С други думи, регистрите се разполагат точно по страниците и колоните и никога не пресичат границите им. Всяка колона има 256 байта, което прави точно 32 регистъра. Регистрите наред с байтовете се явяват „думи“ (word) за процесора, тъй като може да се манипулират с микроинструкции. Създателите на Z80, 6502, че дори и Intel би трябвало да завидят на подобни 64-битови операции.

Страница 0 от паметта RAM на компютъра е предназначена по принцип за приложни програми с капацитет от 4,096 байта. Страница 1 пък е за потребителските данни, чиито 4,096 байта образуват капацитет от 512 регистъра. DOS може да се разположи както в страница 0, така и в 1 и заема 2,048 байта. Ползва данни само от колона 76 и 77, така че не се създава конфликт с потребителската информация. Асемблерът окупира целия компютър (паралелно с DOS) и не оставя място за потребителски софтуер и данни.

Самият процесор има няколко еднобитови флагове, както и едно-, дву- и осембайтови вътрешни регистри, представляващи важни инструменти в неговата работа. Те естествено нямат адреси и са нещо съвсем различно от разгледаните по-горе регистри на паметта RAM.

В компютрите от серията 400 съществуват три групи инструкции. Първата група – т.нар. Group A Instructions, се състои фактически от макросите. Всеки от тях има специфичен код в границите от 0 до 127. В Compucorp, както и в мнозинството от миникомпютрите по онова време, е възприет осмичният запис на числата, така че изписването на кодовете на макроинструкциите е от 0 до 177. Следващите са Group B Instructions, изпълнявани директно от микропроцесора и затова в терминологията на COMPUCOR се водят като микроинструкции. Техните осмични кодове са от 200 до 377. Третата група Group C Instructions са също микроинструкции с кодове от 0 до 177. Превключването на един флаг в микропроцесора определя дали поредната инструкция с код от 0 до 177 да се интерпретира директно от микропроцесора, или трябва да мине през клавиатурния дешифратор и интерпретатор (специализирана част от системния софтуер в ROM) и да се разпознае и изпълни като макрос (чрез задействане на съответните части от системния софтуер в ROM в правилната последователност). Инструкциите от група B са винаги микроинструкции (най-старшият бит на техните кодове е винаги 1) и при тях разпознаването се извършва директно от микропоцесора в рамките на изпълнението им.

Една част от макросите са еднобайтови, състоящи се само от кода на инструкцията. Подразбира се какъв ефект има тяхното изпълнение върху паметта на компютъра и този ефект не може да се модифицира. Друга част се състоят от два или повече байта. Първият байт е кодът на инструкцията, а останалите – операнди за макроса. Някои от операндите може да бъдат непосредствени константи, а други – адреси в паметта, където са разположени фактическите данни. Някои от константите се явяват данни за макроса, а други означават една или друга модификация в изпълнението на инструкцията. Най-сложни и обемисти са макросите за вход и изход, както и тези за функциите на DOS.

Ето как би изглеждал един малък фрагмент от програма, въведена от клавиатурата на компютър Compucorp 445 и започваща от двадесетия байт на страница 0 (за да се улесни изложението се използват правоъгълни скоби за означенията на бутоните от клавиатурата). Най-напред превключвателят RUN / STEP / LOAD трябва да бъде установен на RUN и със следната поредица от бутони **[Jump$_{пп}$] [0] [2]** се задава желаният начален адрес, иначе програмата може да се окаже позиционирана на друго място. След това се превключва на LOAD и от клавиатурата се набират последователно: **[5] [0] [ST$_{пп}$] [+] [1] [2] [HALT] [Jump$_{пп}$] [0] [2]**. С това фрагментът е готов (предишното съдържание на десетте байта на страница 0, започващи от адрес 20, ще бъде загубено) и превключвателят RUN / STEP / LOAD се връща в позиция RUN, за да започне изпълнението на програмата. Нейното действие се изразява в следната последователност:

1. Във входния регистър ENTRY (накратко E, както е възприето от COMPUCORP) се образува цифрата 5, като предварително се занулява съдържанието му. Всъщност с бутоните за цифри от клавиатурата се

задейства в случая макрос, с чиято помощ се формират цифрите в съдържанието на E. Един специален байт на адрес 77 350 служи да сигнализира за формиране на нова величина в E и да се запомни позицията на поредно въведената цифра. Входният регистър е типичен пример за осембайтов вътрешен за процесора регистър.

2. Във входния регистър E се добавя цифрата 0. Благодарение на флаговете на адрес 77 350 не само не се унищожава петицата от предходната стъпка, а и нулата се поставя на правилното място в E и така се получава числото 50.
3. С четирибайтовия макрос **[ST$_{nn}$] [+][1][2]** съдържанието на E, което е така сглобеното 50, се добавя към съдържанието на регистър 12, което е всъщност осембайтовата дума на адрес 20 140. Същевременно на байта на адрес 77 350 се променя съдържанието така, че да сигнализира приключване на сглобяването на величината в E. Както числото 50 във входния регистър, така и първоначалното съдържание на регистър 12 се считат за реални числа с плаваща точка (пакетирани десетични реални числа с експонента в двоичен запис). Резултатът от сборуването, който е новото съдържание на регистър 12, е също реално число с плаваща точка. При системите 400 на COMPUCORP всички числа се сглобяват като реални с плаваща точка. Целочислена аритметика на практика не съществува.

   Самото аритметично действие, както и нормализиране на мантисата на E и различни проверки, се извършва от специализираните програми в ROM.

   За правилното осъществяване на аритметичното действие е необходимо първоначалното съдържание на регистър 12 действително да бъде реално число. Ако вместо това там има нещо друго, примерно текст или просто поредица от случайни байтове, събирането може и да не се осъществи. В такъв случай управлението ще се предаде на програмите в ROM за аритметичен контрол, на системния принтер ще се появи гневно съобщение и компютърът нежно ще се приземи, без да блокира или увисва.
4. Операторът HALT представлява просто пауза. В този момент може да се провери новото съдържание на регистър 12, като се отпечата на системния принтер. За да продължи действието на програмата, се използва бутонът **[RESUME]**.
5. Управлението се предава обратно в началото на програмата с трибайтовия макрос **[Jump$_{nn}$] [0] [2]** и цикълът се повтаря. Към регистър 12 ще се добавят нови 50.

Програмата би могло да се отпечата на системния принтер и би се получило следното:

**Листинг на програма, получена от системния принтер**

| Адрес | Код на операцията и евентуални операнди | Индикатор на принтера |
|---|---|---|
| 0 0 2 0 | 0 0 5 | *5* |
| 0 0 2 1 | 0 0 0 | *0* |
| 0 0 2 2 | 1 2 0 | ↓ |
| 0 0 2 3 | 0 2 1 | + |
| 0 0 2 4 | 0 0 1 | *1* |
| 0 0 2 5 | 0 0 2 | *2* |
| 0 0 2 6 | 0 5 6 | |
| 0 0 2 7 | 1 2 6 | *Ц* |
| 0 0 2 8 | 0 0 0 | *0* |
| 0 0 2 9 | 0 0 0 | *0* |

Реалните числа в машините от серията 400 са с мантиса от тринадесет знака (пакетирани BCD[1]), експонента от два знака и последната шестнадесетична цифра (четири бита) се използва за знака на мантисата и експонентата. Допълнителният код за отрицателни числа не се употребява. Мястото на десетичната точка в нормализираните реални числа се счита, че е непосредствено след първата най-старша значеща цифра (а не преди нея, както е в редица други компютри). В машините от серията 400 на COMPUCORP основните аритметични действия могат да бъдат както двоични, така и десетични. При боравене с реални числа (с плаваща точка) обикновено се ползва десетична аритметика, тъй като числата и без друго са BCD.

Не е констатиран дефект в аритметиката с плаваща точка в машините от серията 400 на COMPUCORP. Основните аритметични действия запазват точността на всичките тринадесет знака на мантисата. Дори и най-сложните степенни и тригонометрични функции дават поне единадесет верни цифри на мантисата на резултата. В по-късния компютър Apple II (Правец 82) аритметиката с плаваща точка е петбайтова, при това се срещат някои съмнителни моменти в алгоритмите дори за основните аритметични действия. При IBM PC пък нямаше въобще аритметика с плаваща точка, а такава се осигуряваше в началото от емулатора 8087. Нещата се оправиха чак с появата на аритметичните копроцесори. Все пак в първоначалните варианти на *Pentium* се откри грешка точно в аритметичната част.

Пакетираните BCD се отпечатват на системния принтер с малки трансформации за поставяне на евентуален знак минус на мантисата и експонентата и вмъкване на десетичната точка на вярното място съобразно величината и знака на експонентата. Числата се отпечатват с експонента само ако са прекалено малки или прекалено големи величини. Ако са с обичайна големина, обработката е доста повече, за да се елиминира експонентата. Отпечатването на терминалите обаче изисква малко повече работа, за да се разпакетират

[1] Binary Coded Decimals, което означава, че само цифрите се кодират двоично, а не цялото число.

числата и цифрите им (в т.ч. знакът на мантисата, десетичната точка и възможната експонента със своя знак) да се трансформират в ASCII кодове в правилната последователност. За всички тези операции в ROM има специализирани програми. Много от тези програми са достъпни за програмиста и ги използвахме в лабораторията за значително ускоряване на отпечатването на табулограмите на бавната пишеща машина.

Приложни програми, съставени от макроси, най-удобно се разполагат в паметта на страница 0. За тяхното редактиране, манипулиране, управление и изпитване съществуват няколко бутона на клавиатурата на по-старите модели като **[Jump$_{nn}$]**, **[Branch$_{nn}$]**, **[HALT]**, **[RESUME]**, **[LIST PROG]**, **[INSERT]**, **[BACKSPACE]** и др. Потребителските данни пък се разполагат на страница 1, тъй като съществуват удобни бутони за оперирane с тях като реални числа в осембайтови регистри: **[ST$_{nn}$]** (за копиране на входния регистър E в съответния на страница 1) и **[RCL$_{nn}$]** (за копиране на съответния регистър от страница 1 във входния E). Освен това има и редица макроси като **[EXCH$_{nn}$]** (съответният регистър от паметта на страница 1 и входният E си разменят съдържанието), които може да не са изведени на бутони (като в модела COMPUCORP 445). Някои от вградените в системния софтуер в ROM математически и статистически операции и функции са изведени на клавиатурата с бутони (линейна регресия при модела 445, тригонометричните функции при 425, логаритми и степенни функции и при двата модела), но повечето не са. Функциите на липсващите бутони може да се заместят, като при създаването на програмата се използва бутонът **[ENTER CODE$_{nnn}$]** и се въвежда кодът на операцията (от 000 до 177 в осмичен запис).

Първите дванадесет регистъра в колона 77 (т.е. байтовете от адрес 77 000 до 77 137 вкл.) също са достъпни за програмиста, като трябва да се имат предвид някои съществени особености. Преди всичко оперирането с тези регистри е по-бързо в сравнение с тези от страница 1. Освен това голяма част от макросите (т.е. системният софтуер в ROM) използват тези регистри за съхранение на крайните резултати. Последното може ефикасно да се използва в приложните програми, стига да не се допуска конфликт със системния софтуер в ROM. Боравенето с тези регистри е улеснено от специални бутони на клавиатурата, като **[ST$_n$]**, **[RCL$_n$]**, а в модел 425 има и **[EXCH$_n$]**.

Вече бе привлечено вниманието, че в модел 450 няма клавиатура като тази в 425 и 445. Запазени са само няколко „жизнено важни“ бутона в дясната част на клавиатурата на терминала, между които цифрите и **[ENTER CODE$_{nnn}$]**, явно само за аварийни ситуации. Не се предвижда и не се препоръчва изготвяне на приложни програми по начина на работа с по-старите модели 425 и 445. За целта се използва асемблерът, предоставящ значително по-добри възможности за работа.

С натрупването на известен опит в работата с компютрите на COMPUCORP стана ясно, че инструкциите (макросите) от група A са отчайващо бавни. Това се дължи на многостепенния механизъм на дешифриране и разпознаване на макроса, което е очевиден рецидив от калкулаторната епоха. Наличието на DOS и асемблер подсказваха вярната посока на развитие. Само с макроси един прогностичен вариант по демографски въпроси отнемаше около

седем часа работа на компютъра. След като преработихме програмите изцяло с група B и C инструкции и възприехме двойно буфериране на данните за печат, се оказа, че същата работа приключва за около един час, като през цялото време пишещата машина печата таблиците. Последното показва, че при такава бавна периферия по-висока скорост не би могло да се постигне и е безсмислено да се търси. Освен това времето за изготвяне на програмите се оказа дори и по-кратко благодарение на системността на работа, осигурявана от дисковата операционна система.

Работата с асемблер на ниво, съвсем близко до процесора, фактически с машинен код, е стъписващо начинание поради многообразието от технически подробности и условности. Освен това е необходима подробна техническа документация, която по онова време в България беше определен и непреодолим дефицит. Трябва да имаш и отзивчиви колеги, готови да се притекат на помощ в тежки моменти, каквито приятели имахме в лицето на ЦЛОЗОИ на БАН. Така например в един момент се появиха известни проблеми с четеца на магнитни карти. Понякога картата се „заклещваше“ в устройството, без да бъде прочетена. Изваждането ѝ с пинсети и куки от кламери бе мъчителна и доста рискована работа. Явно имаше дефекти в микроключовете, регистриращи движението на картата между двете крайни положения и превключващи въртящите се водещи гумени валячета в двете посоки. Накрая устройството съвсем отказа и ни „върза ръцете“, тъй като без магнитни карти няма как да се стартира системата и да се зареди DOS. Нуждаехме се от радикално, и то съвсем оригинално българско решение, тъй като не можеше да се разчита на резервни части за ремонт.

Бе забелязано, че при включване на захранването на компютъра и дисковите устройства главите им автоматично се позиционират на началната писта и нулевия сектор под действието на собствената си електроника. В ЦЛОЗОИ изкопирахме стартовата магнитна карта за DOS върху нулевия сектор на една дискета, която придоби специално предназначение и за нищо друго не би трябвало да се използва. Веднага я направихме защитена срещу запис (write-protected), изкопирахме я за всеки случай и на втора дискета и се втурнахме към най-неизвестната (и най-предизвикателната) част от авантюрата. От клавиатурата на терминала на компютъра 450 в лабораторията ни с помощта на клавиша **[ENTER CODE$_{nnn}$]** се набира максимално кратка програмка (около двадесет байта) в машинен код, за да се прочете нулевият сектор върху първите 512 байта от страница 0 на RAM. С бутона **[RESUME]** се стартира програмата и ... се получи! Образът на магнитната карта се появи точно там, където трябва да бъде, и сега с **[Jump$_{nn}$] [0] [0]** и нов **[RESUME]** можеше да се стартира системата. Така се отървахме от устройството за магнитни карти на сравнително ниска цена: всеки път трябва от клавиатурата да се набира програмката за прочитане на началния сектор на „златната“ дискета.

По принцип програмите на асемблер в машинен код са много по-обемисти и тяхното изработване е в някаква степен филигранна работа, с изпипване на детайлите до степен на елегантност. Но пък се създава възможност за много по-голяма системност и контрол в развойната дейност, в модулното проектиране и програмиране, при което програмистът може да се изяви със собствен

стил на работа. Така например като първа фаза в разработването на даден проект на лабораторията изглеждаше най-добре да се развие модулът за извеждане на крайните резултати (на пишещата машина във формата на таблици като крайна цел). При това този „изходен“ модул много бързо достигаше окончателния си вид с двойно буфериране на данните за печат. В работната му фаза данните се извеждаха на видеодисплея, за да се пестят време и хартия, и в последния момент с дребна модификация се променяха адресите на устройствата за изход. Втората фаза обикновено беше изготвянето на „входния“ модул за въвеждане на потребителската информация в програмата, представляващ обикновено доста тежка задача, тъй като данните се преписваха от различни статистически издания и беше необходимо да се програмира щателен контрол на въвеждането. Готовият вече изходен модул служеше за верификация на правилното функциониране на входния. С тези два модула контурите на програмата са в общи линии изяснени и остава десертът – по-приятната част по програмирането на алгоритмите за обработка на информацията, моделирането на анализа и финализирането на проекта.

Подобно итеративно изграждане на компютърни проекти бе абсолютно невъзможно с клавиатурно програмиране, и то не само защото системният принтер не позволяваше адекватно документиране на работата и не само защото просто с клавиатурата не би могло да се постигне никаква системност в работата каквито и дискови устройства да се ползваха. Асемблерът и дисковата операционна система се оказаха великолепен инструмент за многофазово модулно изграждане на софтуерни продукти за конкретните цели на лабораторията. На този фон изоставянето на инструкциите от група A и преминаването изцяло към микропрограмиране изглеждаха повече като техническа подробност.

## 5. ОПЕРАТИВНОСТ НА ПРОЦЕСОРА

Входният регистър E има уникална структура, подчертано ориентирана и специализирана към работа с числа с плаваща точка. Както вече бе обърнато внимание, реалните числа в тези модели на COMPUCORP са с мантиса от тринадесет знака и два знака експонента, всичките кодирани BCD. Останалите четири бита от всичките шестдесет и четири се използват за знака на мантисата и експонентата. Освен това към входния регистър са „закачени“ още два полубайта (тетрада), изпълняващи функцията на защита срещу препълване на мантисата отгоре (Guard Digit) и отдолу (Underflow Digit). Тези резервни цифри активно участват във всички аритметични операции, в нормализирането на мантисата, при което се налага разместването на цифрите й нагоре и надолу, и в редица други критични операции.

Освен входния 64-битов регистър E процесорът съдържа още няколко други от особено значение. Вероятно най-функционалният от тях е указателят P, сочещ към адреса на следващата за изпълнение инструкция. Адресите в системата 400 са четиринадесетбитови, което прави възможно адресирането на 16 килобайта. В повечето случаи регистърът P сочи към първия байт след последния операнд (ако се случат такива) на текущата инструкция. При някои инст-

рукции за преход като JUMP и BRCH съдържанието на P се определя от операнда на инструкцията, задаващ адреса на следващата инструкция. Тясно свързан с указателя P е т.нар. P-STACK. Това е участък в колона 77 от паметта (от адрес 77 140 до 77 177 вкл.), където с двубайтови думи е организиран стек на адресите за връщане от подпрограми. Вместимостта на стека е 16 адреса и тъй като системните програми от ROM консумират максимум десет нива, за програмиста остава възможността да съчленява верига до шест подпрограми, една в друга. Обаче и DOS може да използва три нива за подпрограми, така че за приложната програма остават само три, което е малко притеснително. Стекът е Last-In-First-Out, което се осъществява с помощта на специален стеков указател вътре в процесора, напълно невидим отвън. Неговата стойност може обаче да се установи по динамиката в стека. Със стека е свързана една много полезна микроинструкция RCLP, която е своеобразен JUMP към адреса от върха на P-стека, като в хода на прехода този адрес се премахва от стека и така се освобождава едно място (всъщност стековият указател се пренасочва).

Микроинструкциите JUMP и BRCH съществуват с по шестнадесет варианта (код на операцията 220 до 237, вкл. за JUMP, и 200 до 217, вкл. за BRCH). Отделните варианти определят колоната от паметта, към която се предава управлението. А операндът на инструкцията, бидейки еднобайтова величина, определя къде точно в колоната е целта. Инструкцията BRCH е фактически JUMP, но освен това адресът на байта непосредствено след операнда на BRCH се качва на върха на P-стека. След като подпрограмата по такъв начин получи управлението, в края ѝ би трябвало да се намери инструкция RCLP, с която ще се върне управлението непосредствено след инструкцията BRCH. Ето един прост пример (като листинг от асемблера AL400):

| LINE | DEC | OCT | CDE | | SOURCE | |
|---|---|---|---|---|---|---|
| 009 | 2493 | 11275 | 211<br>302 | AA | BRCH BB | Преход към подпрограма с етикет BB |
| 010 | 2495 | 11277 | 231<br>275 | | JUMP AA | Връщане към адрес с етикет AA |
| 011 | 2497 | 11301 | 377 | | NOOP | Нулев оператор |
| 012 | 2498 | 11302 | 330 | BB | CLRE | Занулява се всички битове на E |
| 013 | 2499 | 11303 | 371 | | RCLP | Връщане от подпрограмата |
| 014 | 2500 | 11304 | 377 | | NOOP | Нулев оператор |

В първа колона се намират номерата на редовете в изходния текст (source) на програмата в десетичен запис. Във втора и трета колона са адресите на отделните байтове от програмата съответно в десетичен и осмичен запис. В четвърта колона CDE е самото съдържание на байтовете от програмата, състоящо се от кодовете на инструкциите и евентуалните им операнди. В следващата колона е мнемоничният код на инструкцията на асемблерния език, евентуално с етикет, и в последната – кратък коментар. Както се вижда, асемблерът умишлено изпуска адресите на операндите, тъй като не се предвижда преход към тях, нито позоваване на техните позиции и използването на съответните адреси.

Фрагментът е разположен на страница 0 от паметта, колона 11 и заема седем байта, вкл. един излишен NOOP (нулев, празен оператор с машинен код 377, без да броим празните оператори преди и след програмата) на адрес 11 301. Действието на фрагмента започва с оператора BRCH на адрес 11 275, представляващ преход към подпрограмата с начало на адрес 11 302, маркирано с етикет BB. Операндът (на адрес 11 276) на този BRCH е 302, а кодът на BRCH определя колоната на прехода, в резултат на което указателят P придобива съдържание 11 302 и така се осъществява преходът. Същевременно на върха на стека се поставя адресът на връщане 11 277, сочещ непосредствено след оператора BRCH и неговия операнд.

На адрес 11 302 с инструкцията CLRE (с код на операцията 330) се занулява всички битове на входния регистър. След нея се изпълнява RCLP и се осъществява връщането от подпрограмата съгласно адреса на върха на стека.

Входно-изходните операции интензивно използват стека и това може да се случи по време на изпълнението на разглежданата подпрограма. В резултат на това стекът може значително да се видоизмени. Системният софтуер в ROM на компютъра обаче много внимателно използва стека и стриктно възстановява йерархията след използването му.

Този прост механизъм е основният за задействане на подпрограми, реализиран с микроинструкции. Заслужава да се отбележи, че операндите на микроинструкциите JUMP и BRCH сочат към абсолютния адрес на прехода, а не относително, както е при процесорите 6502 и x86, с използване на допълнителен код за отрицателните числа, означаващи преход назад. Втората особеност е, че мнемониците JUMP и BRCH се асемблират в съответния вариант на код на операцията в зависимост от това, към коя колона се предава управлението (но винаги в същата страница от паметта), и операндът само „уточнява“ дестинацията в рамките на тази колона. Така например използваният BRCH се асемблира в 211, а не нещо друго в диапазона от 200 до 217 вкл., тъй като дестинацията е в същата колона.

Свойствата и функциите на микроинструкциите JUMP и BRCH разкриват, че с тях може да се направи преход само в текущата страница на паметта, в която се намират. Ако се налага по-“дълъг“ преход, какъвто е необходим за директно ползване на софтуерните ресурси в ROM, ще трябва да се използва микроинструкцията RCLP или подобната на нея RCLK с код на операцията 370. В група C съществуват няколко още по-бързи инструкции за преход към конкретни места в ROM, които ще бъдат разгледани по-нататък.

Сред микроинструкциите съществуват и условни JUMP за преход при дадено условие, но няма условни BRCH. В повечето варианти на тези инструкции условието се определя от един байт в микропроцесора, означаван като индексен регистър IX в терминологията на COMPUCORP. Битовете на всеки байт, в т.ч. и на този индексен регистър, се означават като L1, L2, L4 и L8 – за младшите четири бита, и H1, H2, H4 и H8 – за старшите. По такъв начин стойността на всеки байт може да се определи като: L1 + 2*L2 + 4*L4 + 8*L8 + + 16*H1 + 32*H2 + 64*H3 + 128*H4. Асемблерната формулировка на условните преходи е от вида JIF L4, AA, означаваща преход на адрес AA, ако битът L4 е 1.

В противен случай L4 = 0, преход не се извършва. Тази конструкция често се среща за откриване дали отделни битове от паметта са 1 или 0.

Аналогично на JUMP и BRCH, тези условни преходи генерират двубайтов машинен код, първият байт на който е от 240 до 247 в зависимост от това, дали се подлага на тест L1, L2, L4, L8, H1, H2, H4 или H8, а вторият представлява абсолютният адрес на дестинацията. По тази причина и за разлика от JUMP и BRCH преходът при подобни условни построения може да стане само в колоната от паметта, където се намира JIF.

Сходни с JIF-овете са JLNZ и JHNZ с еднобайтов операнд, представляващ абсолютният адрес на дестинацията в същата колона. Преходът при JLNZ се извършва, ако някой от четирите най-младши бита на IX (младшата половина на индекса IX, означавана като IL) е различен от нула. При JHNZ на тест се подлагат четирите най-старши бита на IX (горната половина на индекса, отбелязвана като IH).

Съществува още един, подобен на JIF, JLNZ и JHNZ, условен преход, свързан с IX. Той е от вида JINI I0,AA или JINI I1,AA и сравняването на IX става с байта на адрес 77 340, респ. 77 350. Тези два адреса имат в асемблера символичните имена I0 и I1. Преходът се осъществява, ако съдържанието на IX е различно от това в I0, респ. I1.

Условни преходи има и в зависимост от съдържанието на входния регистър E. Те също са с еднобайтов операнд, представляващ абсолютният адрес на дестинацията в същата колона. Преходът при JMNZ се осъществява, ако мантисата на E не е нула (Not Zero). При JMZP ако мантисата не е отрицателна (Zero or Positive). При JXNZ, ако експонентата не е нула (Not Zero) и при JXZP, ако експонентата не е отрицателна (Zero or Positive). Съществува още и JNGZ, при който преходът се извършва, ако горната защитна цифра (Guard Digit) на входния регистър е нула.

Друг важен регистър на процесора е указателят на адреси D. Той е двубайтов, но само младшите четиринадесет бита се използват. Старшият байт на D се бележи като DH, а младшият – като DL. Този указател е свързан с редица често използвани микроинструкции, между които:

| Код на инстр. | Формулировка на Асемблер | Действие |
|---|---|---|
| 331 | STED | Копира входния регистър E на място в паметта, сочено от D. |
| 332 | RCED | Копира регистър от паметта, сочен от D, във входния E. |
| 333 | XCED | Разменя съдържанието на регистър от паметта, сочен от D, с това във входния E. |
| 321 | STID | Копира индекс IX на място в паметта, сочено от D. |
| 322 | RCID | Копира в IX байт от паметта, сочен от D. |
| 323 | XCID | Разменя съдържанието на IX с байт от паметта, сочен от D. |
| 334 | ICDB | Увеличава D с 1, за да се адресира следващият байт от паметта. |
| 335 | ICDR | Увеличава D с 8, за да се адресира следващият регистър от паметта. |
| 336 | DCDB | Намалява D с 1, за да се адресира предходният байт от паметта. |

| | | |
|---|---|---|
| 337 | DCDR | Намалява D с 8, за да се адресира предходният регистър от паметта. |
| 252 | JDE2 XXX | JUMP на адрес XXX в текущата колона, ако съдържанието на D съвпада с това на адрес 77 364, със символично име D2. |
| 271 | LDDL XXX | Копира XXX в младшите осем бита на D, т.е. в DL. |
| 272 | LDDH YYY | Копира младшите шест бита на YYY в старшите осем на D, т.е. в DH. |
| 273 | LDDC XXX | Копира XXX в младшите осем бита на D. В старшите шест бита се зарежда 077. Използва се за по-бързо адресиране на колона 77 от паметта. |
| | LODD YXXX | Това е квазиинструкция, асемблираща се в съответни LDDL XXX и LDDH 0YY. |
| 361 | XFID | Копира съдържанието на IX в DL. |
| 362 | XFDI | Копира съдържанието на DL в IX. |
| 363 | EXDI | Разменя съдържанието на IX с това в DL. |
| 372 | STKD | Поставя съдържанието на D на върха на стека и съответно се актуализира неговият указател за един адрес в повече. |
| 373 | RCLD | Адресът от върха на стека се копира в D и се актуализира указателят на стека за един адрес по-малко. |

Инструкцията STKD съществено се използва при преходите в друга страница на паметта. Вече бе показано, че това не може да се получи с JUMP или BRCH, които са с периметър на действие една страница от паметта, още по-малко с JIF, JHNZ, JLNZ, JINI и подобни, които имат действие в рамките на единствена колона. Използването на STKD за далечни преходи може да се илюстрира по следния начин (един от няколкото възможни). Да допуснем, че трябва да се пресметне естественият логаритъм на величината във входния регистър E. Съответната програма в ROM се намира на адрес 44 150, далече в страница 2 от паметта. Ще се използва и системна функция за отпечатване на съдържанието на входния регистър E, намираща се в ROM на адрес 51 022. Тази функция отговаря на бутона PRINT ANSWER на по-старите модели компютри от серията 400 и е сърцевината на съответната макрокоманда.

Със следния фрагмент, разположен в колона 1 на страница 0 от RAM, се постига решение на задачата:

| LINE | DEC | OCT | CDE | | SOURCE | |
|---|---|---|---|---|---|---|
| 024 | | | | PA | EQU 51022 | Входна точка в ROM на функцията PRINT. |
| 025 | | | | LN | EQU 44150 | Входна точка в ROM на функцията логаритъм. |
| 026 | 0450 | 01302 | 271 | | LODD AA | Формира адреса на AA в D за окончателно |
| | | | 327 | | | връщане от ROM. |
| | | | 272 | | | |
| | | | 001 | | | |
| 027 | 0454 | 01306 | 372 | | STKD | Поставя го на върха на стека. |
| 028 | 0455 | 01307 | 271 | | LODD PA | Формира адреса на PRINT за второ отпечатване. |
| | | | 022 | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | 272 | | | |
| | | | 051 | | | |
| 029 | 0459 | 01313 | 372 | | STKD | Поставя го на върха на стека. |
| | | | 271 | | LODD LN | Формира адреса на функцията логаритъм. |
| | | | 150 | | | |
| | | | 272 | | | |
| | | | 044 | | | |
| 030 | 0464 | 01320 | 372 | | STKD | Поставя го на върха на стека. |
| 031 | 0465 | 01321 | 271 | | LODD PA | Формира адреса на PRINT за първо отпечатване. |
| | | | 022 | | | |
| | | | 272 | | | |
| | | | 051 | | | |
| 032 | 0469 | 01325 | 372 | | STKD | Поставя го на върха на стека. |
| 033 | 0470 | 01326 | 371 | | RCLP | Преминава на адрес от върха на стека. |
| 034 | 0471 | 01327 | 006 | AA | OVRI | Пауза. |

В началото на фрагмента се намират две асемблерни директиви за дефиниция на етикетите PA и LN, представляващи входните точки в ROM на интересуващите ни подпрограми.

Изпълнението започва с последователно зареждане на стека с входните точки на ROM в обратен ред на изпълнение, като най-напред се поставя адресът на връщане, маркиран с етикета AA. С инструкцията RCLP на ред 33 се преминава към първата входна точка на системния софтуер в ROM (последния зареден в стека адрес). Това е подпрограмата PRINT ANSWER и на системния принтер се отпечатва съдържанието на входния регистър E, т.е. на това, на което се търси логаритъмът. Следва подпрограмата за пресмятане на логаритъма и след нея пак PRINT ANSWER, за да се види резултатът. Всяка функция на системния софтуер е верига от подпрограми, завършващи със съответните RCLP, като от стека се изтегля съответният адрес на връщане. Най-накрая, преди последния RCLP, някъде из програмите на системния софтуер, работата е на приключване и адресът AA е изтикан на върха на стека. С последния RCLP идва редът на този адрес и се осъществява завръщането от системния софтуер. Инструкцията OVRI на това място е от група C, представлява пауза и действа подобно на макроса HALT. Всъщност OVRI е в основата на HALT.

Използването на микроинструкцията OVRI (с код на операцията 006) от група C никак не е случайно, а по-скоро е неизбежно. Директното използване на ресурсите от ROM като подпрограми може да стане само ако клавиатурният режим на компютъра е напълно изключен и кодовете на операции от 000 до 177 се интерпретират директно от процесора като микроинструкции от група C. Нещо повече, в процесора има два еднобитови флага, контролиращи положението, така че операторите от системните програми в ROM да не минават излишно през клавиатурната логика и да се правят напразни опити за разпознаване на макросите от група A.

В практическата ни работа в лабораторията това не беше проблем, тъй като бързо се ориентирахме въобще да не използваме макроси в програмите и

скоро инструкциите от група A минаха в забвение. Така например бързо се ориентирахме към определен модел за ползване на функциите на дисковата операционна система изцяло с инструкции от група B и C, в който се използва механизмът от последния пример, на което ще се спрем по-нататък.

Блокирането на инструкциите от група A и форсирането на кодовете от 000 до 177 да се възприемат директно от процесора, като се прескача клавиатурната логика, стават с поставяне на бит L2 = 1 в един еднобайтов флаг в процесора. Обратно, ако L2 на този байт е нула, в действие влизат макросите и микроинструкциите от група C стават недостъпни. В терминологията на COMPUCORP битът L2 на този флаг се нарича GPCI (General Protection Control Index). Всичко това доста напомня на Protected (Native) Mode в микропроцесорите на Intel от серията x86.

Останалите битове на този флаг също имат определена функционалност. Например L4 = 1, означаван като BINA, прави аритметиката изцяло двоична. В такъв режим сборът 00001001 плюс 00000010 би бил 00001011, което всъщност е 9+2=11 като съдържание на еднобайтови полета в двоичен запис. Но ако BINA = = 0, числата се считат за Binary Coded Decimal и тогава резултатът от сборуването би бил малко по-друг: 00001001 + 00000010 = 00010001, което е пак 11, но като BCD.

Манипулирането с този флаг е праволинейно, но има някои особености. Типичният фрагмент за превключване към микроинструкциите от група C се препоръчва да бъде с подобно съдържание:

| LINE | DEC | OCT | CDE | SOURCE | |
|---|---|---|---|---|---|
| 001 | 0016 | 00020 | 306 | RFGA | Копиране на флага в индекса IX. |
| 002 | 0017 | 00021 | 263 | ORIX L2 | Логическо побитово OR на IX с 0000 0010. |
| | | | 002 | | В резултат L2 = 1, без да се променят останалите битове от IX. |
| 003 | 0019 | 00023 | 316 | WFGA | Копиране на IX обратно във флага. |
| 004 | 0020 | 00024 | 316 | WFGA | |

Тук се забелязва повторният запис на индекса IX във флага, което се налага от факта, че микроинструкцията WFGA е сравнително бавна и отнема малко повече от 80 микросекунди. Така че инструкцията на адрес 0020 ($00024_8$) все още би се интерпретирала като макрос. Повторната WFGA на критичния адрес просто изпълнява профилактична функция това да не се случи. Разбира се, би могло на нейно място да има всякаква друга инструкция от група B (които не са чувствителни по отношение на GPCI), примерно нулевата NOOP (с код на операцията $377_8$), която просто няма друго действие, освен да предава управлението на следващия байт от програмата. За съжаление иначе логичното използване на NOOP може да се окаже източник на конфузно положение. При редактиране на програмите от клавиатурата с премахване, вмъкване или разместване на пасажите байтовете със съдържание $377_8$ (всички битове единици) се считат за незаети и на тяхно място може да се настанят самите програми. Така че байтове $377_8$ биха били елиминирани само при сгъстяване на програмите. По тази причина не се препоръчва използването на NOOP в ситуации като разглежданата.

В практиката на лабораторията никога не сме ползвали редактиране на програми с клавиатурата, а само с асемблера, така че не би могло да има подобна опасност. Въпреки това повече като дисциплина и добър стил на програмиране сме се придържали към тази препоръка.

Обратното възстановяване на действието на клавиатурния филтър и макросите от група А би могло да стане със следния фрагмент:

| LINE | DEC | OCT | CDE | SOURCE | |
|---|---|---|---|---|---|
| 001 | 2495 | 11277 | 306 | RFGA | Копиране на флага в индекса IX. |
| 002 | 2496 | 11300 | 267<br>375 | ZERO L2 | Логическо побитово AND на IX с 1111 1101.<br>В резултат L2 = 0, без да се променят останалите битове от IX. |
| 003 | 2498 | 11302 | 316 | WFGA | Копиране на IX обратно във флага. |
| 004 | 2499 | 11303 | 316 | WFGA | |

Разгледаните примери подсказват, че с индексния регистър IX може да се извършват различни побитови логически и подобни операции. Впрочем в системите от серията 400 на COMPUCORP в общи линии няма друг начин за такива действия.

Оперативността на индексния регистър IX се определя и от редица други особености на процесора. Съществува например един половинбайтов регистер IS (Index Store), тясно свързан с IX и служещ за маневриране със съдържанието на IX. Възможен е и обмен на съдържанието на IX с младшия байт на адресния регистър D. Индексният регистър служи и за обмен на данни с паметта, поточно с онзи байт от паметта, към който сочи адресният регистър D. Използва се още и при такива инструкции, като вече разгледаните RFGA и WFGA, както и при обмен на данни с периферните устройства. Част от този богат арсенал е показан в следната таблица (някои от тях, свързани с адресния указател D, бяха вече разгледани, но за пълнота се повтарят):

| Код на инстр. | Формулировка на Асемблер | Действие |
|---|---|---|
| 240 | JIF L1,XXX | Преход на адрес XXX в същата колона, ако L1 на IX е 1. |
| 241 | JIF L2,XXX | Преход на адрес XXX в същата колона, ако L2 на IX е 1. |
| 242 | JIF L4,XXX | Преход на адрес XXX в същата колона, ако L4 на IX е 1. |
| 243 | JIF L8,XXX | Преход на адрес XXX в същата колона, ако L8 на IX е 1. |
| 244 | JIF H1,XXX | Преход на адрес XXX в същата колона, ако H1 на IX е 1. |
| 245 | JIF H2,XXX | Преход на адрес XXX в същата колона, ако H2 на IX е 1. |
| 246 | JIF H4,XXX | Преход на адрес XXX в същата колона, ако H4 на IX е 1. |
| 247 | JIF H8,XXX | Преход на адрес XXX в същата колона, ако H8 на IX е 1. |
| 250 | JINI I0,XXX | Преход на адрес XXX в същата колона, ако IX ≠ I0. |
| 251 | JINI I1,XXX | Преход на адрес XXX в същата колона, ако IX ≠ I1. |
| 262 | EOIX XXX | Изключващо побитово OR на IX с XXX. |
| 263 | ORIX XXX | Побитово OR на IX с XXX. |
| 264 | LDIL XXX | Зарежда IL с младшите четири бита на XXX. |
| 265 | LDIH XXX | Зарежда IH със старшите четири бита на XXX. |
| 266 | LDIX XXX | Зарежда целия IX с XXX. |

| | | |
|---|---|---|
| 267 | ANIX XXX | Логически AND на IX с XXX. |
| 321 | STID | Копира съдържанието на IX в байт от паметта, сочен от D. |
| 322 | RCID | Копира в IX съдържанието на байт от паметта, сочен от D. |
| 323 | XCID | Разменя съдържанието на IX с байт от паметта, сочен от D. |
| 324 | ADIE | Аритметично добавяне на съдържанието на IL в цифрата на мантисата на E, сочена от IH . При IH = 0 добавянето става в Underflow Digit. |
| 325 | XFIE | Копира съдържанието на IL в цифрата на мантисата на E, сочена от IH . |
| 326 | XFEI | Копира в IL цифрата на мантисата на E, сочена от IH . |
| 327 | EXEI | Разменя съдържанието на IL с цифрата на мантисата на E, сочена от IH . |
| 340 | ICIL | Увеличава IL с 1. Няма пренос в IH (т.е. modulo 16). |
| 341 | ICIH | Увеличава IH с 1 (modulo 16). |
| 342 | ICIX | Увеличава IX с 1. С пренос от IL в IH (modulo 256). |
| 343 | SRCI | Кръгово преместване на битовете на IX с една позиция вдясно. |
| 344 | DCIL | Намалява IL с 1. Няма пренос от IH (modulo 16). |
| 345 | DCIH | Намалява IH с 1 (modulo 16). |
| 346 | DCIX | Намалява IX с 1. С пренос от IH в IL (modulo 256). |
| 347 | SLCI | Кръгово преместване на битовете на IX с една позиция вляво. |
| 360 | CLIX | Занулява всички битове на IX . |
| 361 | XFID | Копира съдържанието на IX в DL. |
| 362 | XFDI | Копира съдържанието на DL в IX. |
| 363 | EXDI | Разменя съдържанието на IX с това в DL. |
| 364 | EXIL | Разменя съдържанието на IL с това в IS. |
| 365 | EXIH | Разменя съдържанието на IH с това в IS. |
| 367 | EXIX | Разменя съдържанието на IH с това в IL. |

Като пример за функционалната връзка между индексния регистър IX и входния E да разгледаме следния програмен фрагмент, при който се получава абсолютната стойност на величината във входния регистър E:

| LINE | DEC | OCT | CDE | SOURCE | |
|---|---|---|---|---|---|
| 145 | 2537 | 31351 | 360 | CLIX | Занулява всички битове на IX . В този момент IH сочи към най-младшите четири бита на E, съдържащи знаците на мантисата и експонентата. |
| 146 | 2538 | 31352 | 326 | XFEI | Копира в IH тези четири бита. |
| 147 | 2539 | 31353 | 267 | ANIX 376 | Нулира бит L2, който представлява знакът на мантисата и по такъв начин съдържанието на E става неотрицателно число. По-ясната асемблерска формулировка би била ZERO L2. |
| | | | 376 | | |
| 148 | 2541 | 31355 | 325 | XFIE | Връща обратно тези четири бита на мястото им във входния регистър. |

Подобно на RFGA и WFGA индексният регистър IX и адресният указател D участват в още няколко инструкции от група B за управление на периферните устройства, както и при боравенето на останалите флагове на процесора.

Следният фрагмент показва някои от тези инструкции в действие по директно отпечатване на терминала на серия байтове от паметта. Данните трябва предварително да се разпакетират в ASCII кодове и разположат в паметта по реда на тяхното отпечатване.

| LINE | DEC | OCT | CDE | | SOURCE | |
|---|---|---|---|---|---|---|
| 001 | 00016 | 00020 | 266 | AA | LDIX 320 | За периферно устройство на адрес 13. За адрес 14 операндът трябва да е 340. IH сочи номера на устройството, IL = 0 за функцията Request Status по протокола RS-232. |
| | | | 320 | | | |
| 002 | 00018 | 00022 | 314 | | OPCB | Поставя контролера в Request Status. |
| 003 | 00019 | 00023 | 305 | | IPDB | Получава в IXсъстоянието на устройството. |
| 004 | 00020 | 00024 | 242 | | JIF L4,AA | Ако периферното устройство дава „заето“, повтаря проверката. |
| | | | 020 | | | |
| 005 | 00022 | 00026 | 266 | | LDIX 322 | IH сочи номера на устройството, IL = 2 за функцията Write Data по протокола RS-232. |
| | | | 322 | | | |
| 006 | 00024 | 00030 | 314 | | OPCB | Поставя контролера в Write Data Status. |
| 007 | 00025 | 00031 | 322 | | RCID | Извлича от паметта поредния байт за отпечатване, сочен от указателя на адреси D. |
| 008 | 00026 | 00032 | 315 | | OPDB | Отпечатва се. |
| 009 | 00027 | 00033 | 334 | | ICDB | Увеличава D с 1, за да сочи към следващия байт от данните. |
| 010 | 00028 | 00034 | 252 | | JDE2 AB | Напуска се цикълът, ако е достигнат краят на серията данни, сочен от D2. |
| | | | 040 | | | |
| 011 | 00030 | 00036 | 220 | | JUMP AA | Повтаря процедурата за следващия байт от серията. |
| | | | 020 | | | |
| 012 | 00032 | 00040 | | AB | ..... | |

За правилното функциониране на този фрагмент е необходимо да се добавят още няколко инструкции за управление на системата от хардуерни прекъсвания на компютъра, така че да не възникват конфликти с останалите периферни устройства. Използвахме в лабораторията подобни конструкции за извеждане на данни на видеодисплея (не си струва това да се прави за сравнително бавния принтер LA-34). Резултатът беше много ефектен с мигновеното запълване на екрана с таблици данни.

Управлението и контролът на хардуерните прекъсвания в системата е деликатна работа. Сред инструкциите от група B съществуват две INTD и INTE, служещи за тази цел. Използването на INTD затваря системата от прекъсвания и от този момент нататък всички микроинструкции се изпълняват като една, без никакво прекъсване. Докато не се изпълни инструкцията INTE, освобождаваща прекъсванията. Хардуерните прекъсвания може да се управляват и само за отделните периферни устройства.

Съществуват и няколко важни инструкции от група B за боравене с входния регистър, подобни на разгледаните за D и IX. Широко използвани са RCED, STED и XCED за боравене с 64-битовите регистри от паметта (обмен на данни с входния регистър E), към които сочи указателят на адреси D. Интерес представляват SHLE и SHRE за преместване на цифрите на мантисата на входния регистър една по една наляво или надясно, като в това движение участват и резервните цифри Guard Digit и Underflow Digit. Тези операции имат съществено участие при нормализиране на мантисата и стриктен контрол на грешките от закръгляне на числата. При решаването на подобни деликатни задачи участват още две микроинструкции ICEX и DCEX за увеличаване или намаляване на експонентата на E с единица, както и разгледаните вече ADIE, XFIE, XFEI и EXEI, и различни условни преходи в зависимост от съдържанието на входния регистър. Един компютър обаче не би могъл да „диша“ без инструкция като CLRE, занулляваща всички битове във входния регистър E.

## 6. КОЛОНА 77 ОТ ПАМЕТТА НА КОМПЮТЪРА И ИНСТРУКЦИИТЕ ОТ ГРУПА C

Последните 512 байта от паметта на компютъра са RAM, интензивно използван от системните програми в ROM. Първата половина от този ресурс на адреси от 76 000 до 76 377 (колона 76) се използва от инструкциите от група A и е на разположение на програмиста, ако не използва макроси. Има 40 байта (на адрес от 76 310 до 76 357), които трябва да се заобикалят, тъй като процесорът ги използва за управлението на системния принтер и периферните устройства.

Втората половина, колона 77 (от адреси 77 000 до 77 377), е по-интересна от софтуерна гледна точка. Този сектор от паметта се разглежда по-скоро съставен от 32 осембайтови регистъра, отколкото 256 байта. Както бе посочено в т. 4., първите дванадесет регистъра имат пряко отношение към работата на макроинструкциите и множество математически, статистически, финансови и други сметки, осъществявани с помощта на съответните програми от системния софтуер в ROM. Много от междинните и крайните резултати се разполагат в тези регистри. За улеснение на потребителя на клавиатурата са изведени дори специални бутони за боравене с тези регистри.

При използване само на инструкции от група B и C тези регистри са свободни за използване от програмиста, но ако в програмите директно се използват ресурсите от ROM, необходимо е внимателно да се проучи кои от тези регист-

ри се въвличат в съответните програми и какви резултати се появяват там с оглед избягването на конфликти.

Следващите четири регистъра (на адрес 77 140 до 77 177 вкл.) са стекът на адресите и тази зона трябва също внимателно да се заобикаля.

Най-интересни са шестнадесетте регистъра след адрес 77 200. Те имат символични асемблерни имена R20 до R37 и съществуват съответни микроинструкции от група C за опериране с тях с формата на STR R20, RCR R20 и XCR R20. Тези микроинструкции са еднобайтови, при тях не се използва указателят за адресите D и затова са изключително бързи при изпълнение. Така например инструкцията STR R20 копира входния регистър E в R20, RCR R20 прави обратното – съдържанието на R20 се копира в E, а XCR R20 просто разменя съдържанието на E и R20.

Някои от тези регистри също се използват от системното осигуряване в ROM, но това е добре документирано. Регистърът R27 например участва във всички аритметични действия с плаваща точка, на което трябва да се спрем по-подробно. Регистърът R26 съхранява предходното съдържание на E, след като в E се въведе число от клавиатурата. Регистърът R30 съдържа случайно число, което се модифицира всеки път, когато компютърът изчаква команда от потребителя от системната клавиатура. Регистрите R23, R24, R25, R31, R32 и R33 се използват за различни математически, статистически и финансови и други сметки, осъществявани с помощта на съответните програми от системния софтуер в ROM. Регистрите R20, R21 и R22 пък се използват при наличието на скоби във формулите (съставяни със средствата на група A), с което се определя съответен приоритет на аритметичните действия. Ако се работи само с микроинструкции, тези регистри се оказват напълно свободни и са на разположение за потребителските програми. Същото важи и за R26, ако данните се въвеждат само от терминала, което практикувахме в лабораторията. Когато не ползвахме генератора за случайни числа, в употреба влизаше и R30.

Последните четири регистъра са още по-особени. Осемте байта на R34 имат отделно предназначение, като младшите четири имат символични имена I0, I1, I2 и I3 (по реда на нарастване на адресите). В предходната точка бе показано как при микроинструкциите JINI от група B се въвличат I0 и I1. Следващият R35 също е разпределен на осем байта и младшите четири са съответно със символични имена I4, I5, I6 и I7.

В група C има високоскоростни микроинструкции за боравене с тези байтове във формата STI I0, RCI I0 и XCI I0. Инструкцията STI I0 копира индексния регистър IX в I0, RCI I0 обратно копира I0 в IX и XCI I0 разменя съдържанието на IX и I0. Аналогично е и за останалите I1, I2, ...I7.

Повечето от байтовете на R34 и R35 имат някакво функционално предназначение за програмите в ROM. Например I4 съдържа различни индикации, когато се набира някакво число от клавиатурата на компютъра (но не и от терминала). Байтът на адрес 77 347 пък съдържа указание с колко знака след десетичната точка да се печатат числата на системния принтер и по принцип няма особен смисъл неговото съдържание да се засяга и изменя от потребителски програми с извеждане на данни на терминал или пишеща машина. Пък и няма символично име и не може да участва в бързите микроинструкции. Напълно

неангажиран от ROM остава само I0. Функциите от ROM запазват първоначалното съдържание на индексния регистър IX в I3 и при излизане от ROM го възстановяват обратно в IX, което може да се използва при програмирането. Без особени ограничения може да се ползва и I1, тъй като в него се съхранява кодът на последния натиснат клавиш, което в общи линии е безполезна информация за приложните програми.

Регистрите R36 и R37 са разделени на осем двубайтови полета със символични имена D0, D1, ... D7, съответни на указателя на адресите D. Налице са и съответните микроинструкции от група C за боравене с тях във формата STD D0, RCD D0 и XCD D0. Инструкцията STD D0 копира съдържанието на указателя на адресите D в D0. Инструкцията RCD D0 пък прави обратното – копира съдържанието на D0 в указателя на адресите D. Инструкцията XCD D0 разменя съдържанието на D и D0. Повечето от тези полета са плътно ангажирани от програмите в ROM и не бива да се засягат от потребителските програми. Напълно свободни са само D1 и D2. В предходната точка се привлече вниманието върху микроинструкцията JDE2 от група B, при която се ползва съдържанието на D2. Функциите от ROM запазват първоначалното съдържание на адресния указател D в D3 и при излизане от ROM го възстановяват обратно в D, което също може да се използва при програмирането.

Ето един модел за ползване на функциите на DOS изцяло с микроинструкции и съответни елементи от колона 77 на паметта, към което се ориентирахме в практическата ни работа в лабораторията при изготвяне на компютърните ни програми:

```
DS STD   D3     Запазва се указателят D. С връщането от DOS това съдържа-
                ние на D ще се възстанови.
   LDIX  H1     За да не реагира системният принтер на съобщенията от DOS.
                Иначе LDIX 000.
   STI   I6
   STI   I7
   STD   D0
   STD   D6
   LDIX  H4     За дисковите устройства на адрес 4. За адрес 5: LDIX H1,H4.
   LDDC  R33
   STID         Записва IX в началния най-младши байт на R33.
   LODD  41205  Втори входен адрес за ROM, откъдето ще се задвижи и DOS.
   STKD
   LODD  62323  Първи входен адрес за ROM.
   STKD
   RCLP
```

След завръщането от ROM и DOS в IX (с копие в I3) се намира съобщение от DOS, ако има такова (иначе IX = 0). Използването на този фрагмент става по следния начин:

```
   LODD  YYXXX  Указателят за адреси трябва да сочи към параметрите на DOS
                операцията.
```

```
BRCH   DS      Отваря файл с параметри, сочени от D.
OPNF           Символично име на функцията OPEN FILE на DOS.
```

След завръщането от ROM и DOS е уместно да се провери съдържанието на IX и ако в него има нещо различно от нула, да се установи кое го е предизвикало.

Друг пример:

```
BRCH   DS
CLAF           Символично име на функцията CLOSE ALL FILES на DOS.
BRCH   DS
MON            Прекратява се действието на програмата и се стартира
               MONITOR.
```

В този пример не е необходимо да се настройва указателят на адреси D, тъй като и двете операции CLOSE ALL FILES и MONITOR не изискват параметри. С подобен фрагмент ни завършваха програмите, изработвани в лабораторията.

Регистърът R27 участва активно във всички аритметични действия с плаваща точка. Съответното програмно осигуряване се намира в ROM, но в процесора е предвидено директно обръщение с помощта на специализирани микроинструкции от група C, с което се избягва многостепенната и малко претрупана конструкция, разгледана в т. 5. Това е направено с цел аритметичните действия като често срещани операции да се извършват с максимална скорост и не е съвсем логично обръщението към ROM за такава цел да се окаже тясно място. Тези микроинструкции са:

| Код на инструкцията | Формулировка на Асемблер | Действие |
|---|---|---|
| 004 | FADD | E + R27, като резултатът отива както в E, така и в R27. |
| 010 | FSUB | E – R27, като резултатът отива както в E, така и в R27. |
| 012 | FMLT | E * R27, като резултатът отива както в E, така и в R27. |
| 014 | FDIV | R27 / E, като резултатът отива както в E, така и в R27. |

Следва пример за използване на тези микроинструкции за получаване на величината във входния регистър E на трета степен:

```
LINE  DEC  OCT    CDE   SOURCE
151   256  01000  127   STR R27   Копира E в R27.
152   257  01001  012   FMLT      Във входния регистър E и R27 се получава
                                  изходната величина на квадрат.
153   258  01002  151   RCR R31   Възстановява в E първоначалното съдър-
                                  жание, запазено от FMLT в R31.
152   257  01003  012   FMLT      Получава се третата степен едновремен-
                                  но в E и R27.
```

Микроинструкциите за аритметични операции са фактически много бързи „суперджъмпове“ – преходи в ROM. Така например с такъв „суперджъмп“ инструкцията FDIV предава управлението на адрес 40 014, съвсем в началото на ROM. На този адрес има „обикновен“ JUMP към адрес 47 074, където е факти-

чески началото на програмата за аритметично деление на две реални числа с плаваща точка – делимото в R27, а делителят в E. Тези „суперджъмпове“ напомнят за прекъсванията (interrupt) на IBM PC – както хардуерните с адреси 00h до 1Fh, така и софтуерните 20h до 3F, обслужващи предимно MS DOS.

В група C съществуват още шест микроинструкции, които представляват градивните елементи на аритметиката с плаваща точка и които интензивно се използват от съответните програми в ROM:

| Код на инструкцията | Формулировка на Асемблер | Действие |
|---|---|---|
| 110 | ADEM | Прибавя аритметично мантисата на R27 към тази в E. |
| 111 | SBEM | Изважда аритметично мантисата на R27 от тази в E. |
| 112 | ADEX | Прибавя аритметично експонентата на R27 към тази в E. |
| 113 | SBEX | Изважда аритметично експонентата на R27 от тази в E. |
| 114 | ADRP | Прибавя аритметично IL пъти мантисата на R27 към тази в E. |
| 115 | LDRE | Зарежда всички битове на E с единица и занулява резервните цифри (Guard и Underflow). |

Аритметичните инструкции FADD, FSUB, FMLT и FDIV не са единствените специализирани оператори за бързо предаване на управлението на програмите в ROM. Повечето кодове от 000 до 017 са такива, като управлението се предава на колоната 40 от паметта и точният адрес се определя от кода на самата инструкция. Например FDIV има код 014 и с нея управлението се предава на адрес 40 014. В т. 4 се спряхме на микроинструкцията OVRI, действаща като пауза. Нейният код е 006 и предава управлението на адрес 40 006. Там се намира JUMP към същинската програма на адрес 40 176, но това е вече друга работа.

Ето какво представляват тези експресни инструкции за преход:

| Код на инструкцията | Формулировка на Асемблер | Действие |
|---|---|---|
| 000 | OFLO | В случай на препълване при аритметични действия с прекалено големи или прекалено малки числа. |
| 001 | ERR | В случай на препълване при опит за деление на нула, при опит да се получи логаритъм от отрицателни числа и подобни абсурдни аритметични и алгебрични ситуации. |
| 002 | KEYI | Клавиатурният дешифратор и интерпретатор на макроинструкциите. |
| 004 | FADD | Събиране. |
| 006 | OVRI | Пауза. Тази микроинструкция е в основата на функцията на бутон HALT на системната клавиатура, запазен и при модела COMPUCORP 450, но означен като CLEAR и с малко разширени функции. |
| 010 | FSUB | Изваждане. |
| 012 | FMLT | Умножение. |
| 014 | FDIV | Деление. |

| | | |
|---|---|---|
| 016 | RSTI | Master Reset за възстановяване на системата при аварийни ситуации. Тази микроинструкция е в основата на функцията на бутон RESET на системната клавиатура, запазен и при модела COMPUCORP 450. |

Тези експресни преходи в ROM имат и една друга особеност, която не се среща при останалите микроинструкции. Кодовете 000 до 177 вкл. се възприемат директно от процесора като инструкции от група C, ако флагът GPCI е включен, както това е разгледано в т. 5. При това положение, ако се изпълни някоя от горните микроинструкции за бърз преход в ROM, оказва се, че още един флаг в процесора се включва, т.нар Supervisor. От този момент нататък всички микроинструкции с код 000 до 177 вкл. се изпълняват като от група C независимо от състоянието на флага GPCI. Единствената възможност да се изключи супервайзърът е да се мине през инструкция RCLK, чието действие е като RCLP, но и загасва супервайзерът. Този флаг на процесора не е достъпен по друг начин.

## 7. АСЕМБЛЕР AL400

Асемблерът AL400 вероятно отбелязва финалът в развитието на серията 400 на фирмата COMPUCORP и с него последният модел 450 придобива завършен вид на компютър. Този софтуерен комплекс се оказа мощно средство за развойна дейност и в лабораторията бе използван като основен инструмент за създаване на програмното осигуряване за демографския анализ. Чувстваше се обаче липсата на софтуер за DEBUG и възможности за трасировка на програмите.

Асемблерът е комплект от софтуерни програми в програмната библиотека на специална дискета, образуваща отделен том с непроменимо име ASSMBLR. При работа с него дискетата трябва да бъде постоянно в едно от дискетните устройства. В другите устройства може да има потребителски томове, в които да се съхраняват изработените програми, готови за изпълнение (object code според терминологията на COMPUCORP[1]), както и файловете с изходния им текст (source code). Във всеки отделен момент обаче асемблерът работи само с един потребителски том. При смяна на потребителския том се налага да се рестартира асемблерът, за да разпознае новата конфигурация. Възможно е да се работи и без потребителски том, при което системната дискета изпълнява функцията на такъв, макар че там свободното пространство е доста ограничено.

Активността на асемблера се познава по неговия промпт (::) и най-често се постига чрез монитора на дисковата операционна система, като се стартира програмата ASSMBLR. Възможно е стартирането на асемблера да става и със специално изготвена за целта DOS LOAD & GO магнитна карта. Оттук нататък работата продължава изцяло в диалогов режим, като асемблерът реагира адекватно на поставените задачи, вкл. и със съобщения за евентуално неправилни, непълни или погрешни команди, зададени от потребителя. Асемблерът

[1] В системата не съществува object code в смисъла на IBM и Microsoft, нито линкери и подобни междинни звена.

прихваща и съобщенията от дисковата операционна система и излъчва само онези от тях, касаещи работещия с компютъра. Например няма смисъл да се излъчва съобщение за край на файл, тъй като това е проблем на асемблера, при това лесен за преодоляване. Ако липсва обаче някакъв файл на дискетите, това може да е проблем на потребителя и в такива случаи асемблерът ще съобщи за това.

Първата стъпка при започване на работа на асемблера е да се посочи потребителският том, който ще се използва в сесията.

Периферните устройства, участващи в този диалог, може да са различни и в движение да се променят. При първоначалните фази на изготвяне на програмите най-добре се работи на VDU терминала. На по-късен етап, особено при тестване на програмите, отпечатване на листинги и търсене на дефекти, може да се включи и печатащо устройство. В това отношение за нас беше голямо улеснение да разполагаме със сравнително бърз принтер терминал като LA-34 (пишещата машина IBM е прекалено бавна за тази цел).

Самият асемблер представлява елементарен редови редактор, подобен на EDLIN при персоналните компютри и XEDIT при различните OS на IBM (по онова време екранните редактори бяха изключителна рядкост) плюс компилатор. Асемблерният език е сравнително прост и напълно отговаря на стандартите от седемдесетте години на миналия век. За съжаление в езика няма макроконструкции, но има други впечатляващи неща, като създаването на многодетайлизирана таблица (cross reference) на дефинициите на етикетите и символичните имена с техните адреси и съдържание, както и пълен списък на местата в програмите на позоваването към тях. Символичните имена са като R27, D0, I1, означаващи елементи от колона 77 на паметта. Етикетите се задават от програмиста и тяхното съдържание може да се определя явно с директивата EQU или пък да представлява адресът, на който са позиционирани.

Диалогът с асемблера се води с команди от работещия, между които:

| Команди | Действие |
|---|---|
| INPUT | За създаване на файл (source module) с изходния текст на програмите и за въвеждане на данни в него. |
| EDIT | За коригиране съдържанието на source module с вмъкване на нови пасажи, изтриване или преместване на стари и т.н. Старото съдържание на source module може да се запази в архивни файлове. |
| ASSMBLE | За компилация на source module или група от такива. Продукт на компилацията може да бъде листинг с машинните адреси, таблица на етикетите и самият изпълним код (object module по терминологията на COMPUCORP). Листингът от поредната компилация автоматично се съхранява в отделен файл LIST FILE, а изпълнимият код – в OBJECT FILE. |
| OBJECT | Копира изпълнимия код от OBJECT FILE във файл на потребителски том. |
| DEVICE | За дефиниране на устройствата за диалог с асемблера. |
| DELETE | За премахване на source module от системната дискета. |
| RELIST | За отпечатване на LIST FILE, т.е. листинга от последната компилация. |

| | |
|---|---|
| LOG | За извеждане каталога на всички source modules от системната дискета. |
| LISTSM | За отпечатване съдържанието на даден source module. |
| RESTART | За да продължи работата с асемблера след смяна на потребителски том. |
| STATUS | За извеждане на състоянието на системата. Това включва: адресите на устройствата за диалог, името на последно компилирания source module (закупирал по такъв начин LIST FILE и OBJECT FILE), както и името на активния потребителски том. |
| COPYSM | За копиране на source file от един в друг том. |
| MONITOR | За прекратяване работата с асемблера и преминаване към монитора на DOS. |
| OPTIONS | Нещо подобно на обичайните днес HELP, съдържащо командите за асемблера и кратко пояснение за тяхното действие. |
| SMERGE | За сливане на няколко source modules в един |

Създаването на файлове с изходния текст на програмите започва с командата INPUT и продължава с многократни EDIT, докато се постигне окончателният вариант. Междувременно с командата ASSMBLE се изпитват поредните варианти и получените изпълними кодове може да се подложат на практически тест. Съществено улеснение в този процес е листингът от компилацията и таблицата на дефинициите на етикетите с техните адреси, както и адресите на позоваването към тях.

Структурата на отделните редове на файлове с изходния текст (source modules) е съвсем проста и се състои от четири полета с фиксирани позиции: етикет с два буквени знака, мнемоничен код на инструкцията с четири знака, операнди и незадължителен коментар с променлива дължина. Възможни са и редове, запълнени изцяло с коментар. Редакторът следи дали се запълват правилно редовете (при изпълнението на командите INPUT и EDIT) на тези файлове и реагира незабавно при допускане на грешка, така че не се налагат излишни компилации за откриване на повечето печатни грешки.

Компилацията на source modules става с командата ASSMBLE, като се посочат един или няколко source modules за участие в процеса. Някои от тях може да служат само с етикетите си (reference only) и това се указва в командата ASSMBLE. Процесът на асемблиране е тристъпков, като на първия пас се събира информация за етикетите и символичните имена. По такъв начин не се появява трудност по асемблиране на етикети, дефиницията на които още не са срещнати в хода на анализа на изходния текст, какъвто проблем години наред имаше при еднопасовите асемблери на Microsoft. Многопасовият турбо асемблер на Borland беше първият, който се справи успешно с тази задача.

Следващите два паса са същинската част на компилацията и дават крайния резултат. В зависимост от опциите на командата ASSMBLE този резултат би могъл да бъде:

- Опция L за създаване на листинг в LIST FILE, който може да се отпечата в хода на компилацията или по-късно;
- Опция W за създаване на изпълним код в OBJECT FILE, който се записва на дискетите като самостоятелна програма със съответното име;

- Опция G за създаване на изпълним код в OBJECT FILE, без да се записва като отделна програма. При наличието на опция W опция G е излишна;
- Опция C за създаване на таблица (cross reference) на дефинициите на етикетите и символичните имена с техните адреси и съдържание, както и адресите на позоваването към тях.

В асемблера съществуват редица помощни инструменти, значително улесняващи създаването на изходния текст на програмите. Възможно е например етикетите и символичните имена да се използват с т.нар. отместване. Например JUMP AA-4 означава да се предаде управлението четири байта по-назад от позицията на етикета AA. Друг пример може да бъде LDDC I7+4, с което в указателя за адреси D се поставя 77 347, равно на адреса на I7 (77 344) плюс четири. Често се използва и т.нар локален етикет за означаване на текущата позиция. Например JUMP *+8 се асемблира в преход на позиция с осем байта по-напред от текущата. Друг пример би бил LODD *-6, при което в указателя D се поставя текущият адрес минус шест.

Много впечатляващо е как AL400 се справя с добре познатата ситуация на „късите джъмпове“. Такъв проблем може да възникне, когато инструкции за условен преход като JIF, JINI и подобни се окажат на такава позиция, че трябва да предадат управлението на адрес вън от текущата колона на паметта. При подобни ситуации асемблерите за персонални компютри от следващите поколения направо катастрофираха и оставяха всичко в ръцете на програмиста. Ето как (във формата на листинг от компилацията) AL400 би преодолял такъв проблем без намесата на програмиста:

```
LINE   DEC    OCT    CDE      SOURCE
005                    AA     ORG 11275
...                           ...
...                           ...
...                           ...
065                           ORG 13200   Позиционира се следващата част от
                                          програмата от този адрес нататък.
066    2944   13200    234    JDE2 AA     OFF-COL COND
                                          JUMP**PATCH*RANGE ERROR*
                        200
...                           ...
...                           ...
...                           ...
                              ORG 14200   С това само се маркира краят на прог-
161                                       рамата.
**PATCH*       14200    252
**PATCH*       14201    204               CONDITION?
**PATCH*       14202    233
**PATCH*       14203    202               JUMP FALSE
**PATCH*       14204    231
**PATCH*       14205    275               JUMP TRUE
```

Адрес 11 275 е маркиран с етикета AA, използван по-нататък в програмата от инструкцията JDE2 AA. Проблемът възниква от това, че тази инструкция се намира в колона 13, а етикетът – в 11. Асемблерът замества тази инструкция с абсолютен JUMP към свободното място след края на програмата (14 200) и там сглобява малка конструкция, за да се справи с положението. Построението се състои в следното. Инструкцията 252, 204 е фактически истинското JDE2, при което се предава управлението на адрес 204 в същата колона 14, ако D = D2. Там е поставен абсолютен JUMP (с машинен код 231) към адрес 275 от колона 11, т.е. точно към етикета AA.

В случай че се окаже D ≠ D2, управлението се предава непосредствено след комбинацията 252, 204, т.е. на адрес 14 202. Там се намира абсолютен JUMP към адрес 13 202, т.е. към мястото, където маскираният JDE2 би трябвало да предаде управлението в този случай.

Както се вижда от листинга, тази операция по маскиране на условните преходи се придружава от редица пояснителни коментари, създадени от асемблера за улеснение на програмиста.

В асемблер AL400 съществуват редица директиви за улеснение на програмирането. Една от най-често използваните е ORG, с която се позиционира следващият за асемблиране машинен код на адрес, задаван от програмиста (както е в горния пример). Родствени с ORG са директивите RORG и BORG. Първата подрежда кода в началото на следващия регистър, а BORG – на следващия адрес, точно делящ се на десет. Директивата RORG е много удобна за асемблиране на данни – числови величини (реални числа с плаваща десетична точка) в тялото на самата програма, така че да попаднат точно в рамките на регистрите. Директивата BORG е преназначена за маркиране на адреси за преход с използване на инструкции от група A.

Често срещана директива също е EQU. С нея се задава стойност на етикет, примерно CR EQU 015. Кодът 015 (в осмичен запис) съответства на ASCII Carriage Return (за връщане на печатащата глава в най-ляво положение). При асемблиране на текстови данни в тялото на програмата така дефинираният етикет CR може да се използва за разграничаването на отделните редове от текста[1].

Директивата EQU може да се използва за създаване на символични имена на отделни участъци от паметта. Например RA EQU 20000 може да се използва по-нататък в програмата като синоним на началния регистър на страница 1. С друга директива ADDR може този етикет да се асемблира в двубайтов адрес във формата на указателя за адреси D.

С директиви като FLT и NENT може да се сглобяват числа. С NENT се имитира число въведено от системната клавиатура. С директивата FLT се асемблират в тялото на програмата реални величини с плаваща точка точно така, както те биха могли да се получат във входния регистър E. Подобно на FLT

[1] За някои принтери се налага след всяко CR да се добави и LF EQU 012, представляващо LINE FEED за преминаване на нов ред при печатане на текста. Други принтери автоматично добавят LF след всяко CR. В някои системи пък е обратното: LF автоматично предизвиква и CR. Обикновено в принтерите има специален ключ за регулиране на това положение и съответен избор.

действа и директивата ASCI за асемблиране на текст в ASCII код в тялото на програмата в правилната последователност на знаците, така че да е готов за отпечатване.

Директивата BSS е за резервиране на област от паметта, така че продължението на програмата минава след тази област. Резервираната област се запълва с нулеви байтове. Обикновено се използва за запазване на място за различни буфери за данни, примерно за отделните записи на файлове или за сглобяване на таблици в хода на изпълнение на програмите.

Директивата ASMB е подобна на съвременните INCLUDE, но посоченият модул се включва не на мястото на директивата, а след последния байт на основния модул.

Директивата ENTP определя началната точка за изпълнение от всяка програма. При липса на такава в дадена програма нейното изпълнение започва с инструкцията от най-малкия й адрес. Тази директива обикновено се позиционира в началото на изходния текст, за да стане ясна логиката на програмата, но това не е задължително.

В изходния текст място може да намерят директивите RPAS и WPAS, определящи съответните пароли за достъп при ползване на програмите.

В асемблер AL400 съществуват и няколко квазиинструкции като разгледаната вече LODD, функциониращи подобно на макросите в съвременните асемблери. Да припомним, че LODD YYXXX се компилира в последователността LDDL XXX и LDDH 0YY и фактически е равностойна и еквивалентна на нея.

## 8. ВМЕСТО ЗАКЛЮЧЕНИЕ

Системата 450 на COMPUCORP бележи края на калкулаторния период на фирмата, след което компанията се ориентира към създаването на малки настолни компютри за персонално ползване, наричани от фирмата desk-top-computers, с възможности за свързване в мрежа като компонент на офис оборудване. По това време на пазара на калкулатори вече са се настанили производители като Hewlett Packard, Casio, Canon, Sanyo, Texas Instruments и др. Калкулаторът HP-35 на Hewlett Packard бързо печели пазара и става шлагер с ниските си цени въпреки пълната липса на програмируемост. Много скоро Texas Instruments и Mostek лансират едночипови програмируеми калкулатори на цени, по-ниски дори от тези за HP-35. Надпреварата в пазара на калкулатори става ожесточена и делът на програмируемите калкулатори от най-висок клас като тези на COMPUCORP заплашително започва да се свива само до научни изследвания и специализирани приложения. Същевременно неимоверно нараства търсенето на елементарни портативни джобни калкулатори с четирите най-прости аритметични действия, т.нар. в Америка four-banger.

При новите условия към края на седемдесетте години на миналия век настъпват радикални промени във фирмата COMPUCORP, вкл. различни разместванията на собствениците. От фирмата се отцепва частта, занимаваща се с компютърни мрежи, която образува самостоятелната компания Retix. Дори са били водени преговори за поглъщане на COMPUCORP от Philips Data Systems,

проявяваща голям интерес в този сектор по онова време. Паралелно с това се налагат и промени в стратегията и фирмата се ориентира към съвсем нови офис системи, базирани на новия и перспективен за това време осембитов микропроцесор на Zilog Z80.Този микропроцесор, по-скоро в неговите съвременни варианти като шестнадесетбитовия Z8080, се среща и днес повече в индустриални контролери, информационни декодери и подобни. Както обърнахме внимание в началото, системите на COMPUCORP, базирани на Z80, представляваха перспективно за този период изделие с добри възможности за приложение в бизнеса. Може само да се гадае защо обаче фирмата не успя да се задържи на пазара. Може би са били подценени графичните възможности на появилия се Apple, на възможностите на проекта на Intel за процесора 8086, на включването на IBM в пазара на персонални компютри и напредъка на Microsoft в сферата на софтуера и невероятното многообразие от софтуерни продукти за прочутия компютър Apple II и впоследствие за IBM PC. Може би COMPUCORP не са успели своевременно да се преориентират от сравнително тясната сфера на научните приложения към ежедневния бизнес.

Така или иначе под напора на бурното развитие на персоналните компютри преди двадесетина години COMPUCORP загубва позициите си като технологичен авангард в този сектор и в края на 1984 г. престава да съществува. Много други известни компании са имали подобна съдба в калкулаторния бизнес като Bowmar, MITS, Cintra, Wang, Tektronix, Sony и др., но, както счита Рик Бенсон[1], загубата на COMPUCORP е трагично събитие за онзи период. Според него специалистите на фирмата са били изпреварили своето време с проектирането на калкулаторния си чипсет, представляващ прототипът на бъдещия микропроцесор в момент, когато не е имало дори и най-бледата идея за подобно понятие.

Историята на COMPUCORP прилича на класически американски уестърн. В края на филма главният герой, изцяло изтъкан от добродетели, бива застрелян в една съвършено нелепа ситуация.

[1] The History of Compucorp, http://www.oldcalculatormuseum.com/d-compucorp.html.

## ПРИНОС В РАЗВИТИЕТО НА ПЕРСОНАЛНИТЕ КОМПЮТРИ: ЛЕГЕНДАРНИЯТ COMPUCORP 450

Резюме

В хода на организирането на Лабораторията за демографски изследвания към катедра „Труд и социална защита“ в Университета за национално и световно стопанство през 1977 г. стана пределно ясно, че не би могло да се осъществява сериозна научна работа по анализа на демографските данни без подходяща изчислителна техника. Още по-малко да се изгражда база данни за българското население в разнообразни разрези, позволяващи достигането на резонни хипотези за бъдещото развитие и програмирането на различни прогностични варианти. Ръководителят на лабораторията проф. Никола Т. Наумов съумя да убеди държавните и партийните авторитети по онова време, че само с молив и хартия не може да се получат желаните научноизследователски резултати по такива парливи проблеми, като демографските, и не може да се очаква сериозна научна подкрепа за държавната политика по населението. В резултат на тези усилия лабораторията бе оборудвана с модерния за времето си настолен компютър COMPUCORP 450.

В студията се разглеждат особеностите на този предтеча на персоналните компютри и е направен опит да се покаже какви тенденции се откриват в развитието на информационните технологии през седемдесетте години на двадесетия век.

## CONTRIBUTION TO THE DEVELOPMENT OF PERSONAL COMPUTING: THE LEGENDARY COMPUCORP 450

Summary

During the organization of the Laboratory for Demographic Research/Labour and Social Protection Department of the University of National and World Economy in 1977 it became quite clear that no serious scientific work on demographic data analysis could be feasible without suitable computer technology. Let alone to develop a multi-dimensional data base for the population of Bulgaria which should facilitate building reasonable hypotheses for the future and verifying different forecasting alternatives. The head of the Laboratory Prof. Nicola T. Naoumov managed somehow to convince party and government authorities of that time that with paper and pencil only it is impossible to derive the desired scientific research products on such sensitive problems like demographic ones and to expect serious scientific support for the government policy on population. The outcome of these efforts was the equipment of the Laboratory with a present-day computer like Compucorp model 450.

In this study the features of such a forerunner of personal computers are examined and it is attempted to reveal what tendencies could be spotted in the development of information technologies during the seventies of the last century.