Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма:Машинско инжењерство

Ниво студија: Основне академске студије

Модул: Информатика у инжењерству

Предмет: Архитектура рачунарских система

Број индекса: 16/2009

Дејан Р. Мутавџић

ARC архитектура

завршни рад

Комисија за преглед и одбрану:

1. **Проф. Др.Јасна Радуловић - ментор**
2. _____ Датум одбране:_____
3. _____ Оцена:_____

У оквиру овог завршног рада кандидат треба да проучи препоручену, као и ширу литературу која се односи на ARC архитектуру. На бази усвојеног знања кандидат треба да обради основне појмове CISC и RISC архитектуре, а нарочито ARC архитектуру, као једну од грана RISC архитектуре, и њене карактеристике.

Препоручена литература:

[1] M.J. Murdocca, V.P. Heuring, *„Principles of computer architecture“,* Prentice Hall International, Inc. pp. 101-125, New Jersey 2000.

[2] Texas Instruments *„MSP430x44xx Family User's Guide",* Vol 3. pp. 41-115, 2006.

Крагујевац, октобар 2013.

ментор:

Проф.Др.Јасна Радуловић

Резиме рада

У овом раду детаљно је обрађена ARC архитектура, као једна од грана RISC архитектуре која се бави креирањем процесора са смањеним скупом наредби и повећањем броја регистара доступним централној процесорској јединици. Када је реч о ARC архитектури, говори се о њеним карактеристикама, структури и скуповима инструкција.

Кључне речи: архитектура, процесор, инструкција.

Abstract

In this work the ARC architecture, one of the branches of RISC architecture that deals with the creation of processors reduced instruction set and increase the number of registers available to the CPU. When it comes to ARC architecture discusses its characteristics, structure and sets of instructions.

Keywords: architecture, processor, instruction.

Садржај

1. Увод

Велико је питање историчара ко је створио концепт модерног рачунара, али већина се слаже да је концепте модерног рачунара створио у деветнаестом веку енглески проналазач Charles Babbage. Његови напори да се створи програмабилни рачунар са технологијом тог времена га је довело до неуспеха , јер масовна производња јединствених механичких делова са ниском толеранцијом није био доступан.[9]

Први компјутери су Colossus. Рачунар је конструисан 1943. године за време Другог светског рата . Начињено је у строгој тајности и користила су га за дешифровање поверљивих немачких порука (под Туринговим надзором). Рачунске операције је обављало 2000 електронских цеви . Рачунар се састајао од улаза за податке, односно 5 рола папира са рупицама на којима је била порука коју је требало дешифровати. Траке папира читале су се оптичким читачем и низ рупица претварао се у електричне импулсе. Електрични су се импулси затим преносили и над њима су се извршавале различите операције. На крају се добила дешифрована порука.[9]

Konrad Zuse је 1942. године конструисао Z3, први релејни рачунар са једноставном контролом рачунарског програма заснованог на бинарном систему.[9]

Амерички физичар Howard Aiken конструисао је 1944. године електромеханички рачунар за аутоматско решавање диференцијалних једначина. То је била велика и компликована машина названа МАРК 1. Рачунар је имао 3300 уграђених електронских цеви и много других делова повезаних са укупно 80 км жице . Било је хиљаду пута брже од најбржег тадашњег механичког рачунара, чиме се и завршила ера механичких машина за рачунање.[9]

Постепено побољшање производне технологије и минијатуризације (транзистора и интегрисаних кола) крајем 20. века, рачунари су постали приступачнији и саставни део цивилизације. [9]

2. Историја рачунара

Компјутери прве генерације (1945-1955), чију су основу чиниле вакуумске цеви (до 20.000 цеви по рачунару), били су огромних димензија и веома скупи. Користила их је углавном војска. Ови рачунари су били јако спори, програмирало се на машинском језику, док су симболички језици (укључујући и асемблер), као и оперативни системи, у то време били непознати . Људи који су радили на тим рачунарима обављали су све послове - од програмирања до одржавања рачунара.[2]

У првој генерацији рачунара, опслуживање рачунарског система било је потпуно препуштено оператеру, који је морао да припреми све што је потребно да се задатак обраде може обавити. Човек је, дакле, имао пуну контролу над рачунарским системом. Оператер је био у могућности да све потребне радње обави на време, јер је систем био спор и извршавао се само један програм, тј . обављао се само један задатак. Може се рећи да је искоришћење рачунарског система, тј. његових најважнијих ресурса - централног процесора и централне меморије - било слабо. Највећи део времена трошио се на послове оператера и улазно - излазне операције, а много мањи део на рад централног процесора. Иако је овакав систем био неефикасан, однос тих временских периода био је у прихватљивим границама због релативно мале брзине самог централног процесора.[2]

Основу рачунара друге генерације (1955-1965) чинили су транзистори, па су рачунари постали мањи, поузданији и јефтинији. Рачунаре друге генерације су, осим војске, куповале и велике корпорације и универзитети. Рачунари су били

смештени одвојено, у посебним собама, које су се делиле у три функционалне целине: улазна соба, централни рачунар и излазна соба. Програмери су писали програме на папиру, већина на програмском језику ФОРТРАН, затим су се ти програми преносили на бушене картице, које су се остављале у соби са улазним рачунаром (input room). Оператор система је, затим, узимао бушене картице и убацивао их у рачунар и то прво картице са преводиоцем ФОРТРАН-а, а потом бушене картице са програмом који треба извршити. Главни рачунар је обављао посао, а резултат се такође добијао на бушеним картицама, које су се преносиле у просторију са резултатима (output room). Овде се много времена трошило на шетаче између разних просторија са бушеним картицама. Оперативни систем као засебан појам још увек није постојао. У даљем развоју се као побољшање уводи пакетна, тј. групна обрада (batch processing), заснована на употреби магнетне траке - уређаја много бржег од бушених картица. При пакетној обради, у улазној соби са пословима сакупља се једна количина сличних програма (на пример, сви програми који захтевају преводилац ФОРТРАН - а), који се помоћу јефтинијег рачунара (нпр. IBM 1401) са бушених картица преносе на магнетну траку. После тога се магнетна трака преноси у собу са главним рачунаром, тј. са моћнијим и скупљим рачунаром, предвиђеним за извршавање самог програма (нпр. IBM 7094). У главни рачунар се учитава посебан програм који је задужен да са траке са пословима програме редом учитава и извршава . Тај програм се може сматрати претком оперативних система. Након извршавања програма, резултати се снимају на другу магнетну траку коју оператер преноси до трећег рачунара, задуженог за пребацивање резултата са магнетне траке на бушене картице. Мањи рачунари (улазно - излазни) нису директно везани за главни рачунар, што значи да раде у off-line режиму (у режиму где нису били у међусобној комуникацији).[2]

У другој генерацији рачунарских система повећава се брзина рада централног процесора, капацитети централне меморије и екстерних (масовних) меморија, појављују се нове и брже улазно - излазне јединице. Програми се пишу на симболичком машинском језику, асемблеру или на вишем програмском језику (ФОРТРАН). Оператер више није у стању да ефикасно опслужује рачунарски

систем, јер су његове реакције сувише споре. Једино решење се могло наћи у пребацивању низа контролних функција са оператера на сам рачунарски систем, тј. на посебне контролне програме. Отуда су функције опслуживача и управљача системом биле подељене између оператера и контролних програма. Ти програми укључују се у одређеним ситуацијама, као што су, на пимер, припрема за извођење програма, контрола улазних и излазних уређаја и разни послови око учитавања програма и припреме за његово извођење. Дакле, у рачунарима друге генерације се разликују две основне врсте програма: контролни и кориснички. Рачунари треће генерације (1965-1980) се праве од интегрисаних кола (IC). Почетком шездесетих година већина произвођача прави две врсте рачунара:

• једну бржу верзију (као IBM 7094) и
• једну слабију (као IBM 1401),[2]

Нови корисници рачунара најпре желе слабије моделе који су јефтинији, док ће им јачи, бржи и скупљи модели бити потребни тек након извесног времена. IBM тај проблем покушава да разреши увођењем класе рачунара IBM систем/360. То је серија компатибилних рачунара различитих снага. Сваки од ових рачунара је погодан и за научну и за пословну примену, па је тиме подела рачунара на ове две врсте нестала. Овај концепт су преузели и остали произвођачи рачунара. Рачунари из серије систем/360 радили су под оперативним системом ОС/360, који је био веома гломазан и препун грешака. Са развојем дисциплине познате под именом софтверско инжењерство (software engineering), уводе се нове функције :

• мултипрограмирање, (multiprogramming)
• вишеструке улазно - излазне (У / И) операције (spool)
• подела рачунарског времена (time-sharing).[2]

Када неки програм чека на резултате улазно - излазних операција, процесор је неискоришћен, па се губи процесорско време. Овај проблем није толико изражен код програма који ретко захтевају улазно - излазне операције (на пример,

научно оријентисани програми), али јесте код пословних програма. Мултипрограмирање је техника којом се постиже боље искоришћење процесора: меморија се дели на партиције у које се учитавају различити програми, тј. послови (jobs). Док неки програм чека на улазно - излазну операцију, процесор може извршавати други програм. На тај начин, ако имамо довољан број програма у меморији, процесор се стално употребљава.[2]

Спулинг (Spool–Simultaneous Peripheral Operation On Line) јесте техника која омогућава да се недовољна брзина улазно - излазних уређаја компензује употребом брзих уређаја као што су траке, нарочито дискови. На тај начин се омогућава истовремено извршавање више улазно - излазних операција. Брзи уређај прихвата све са улаза, а затим се улаз ка процесору реализује са брзог уређаја. Улаз се реализује пребацивањем садржаја бушених картица на диск (траку) помоћу посебног уређаја, а без коришћења процесора. То значи да се диск паралелно пуни новим пословима док процесор извршава програме у меморији. Када један програм заврши рад, процесор на његово место може учитати други програм са диска. Све што се шаље на излаз, прво се преноси на брзи уређај, а затим са њега на споре периферне уређаје. Подела времена (time-sharing) јесте техника која омогућава да сваки корисник ради с рачунаром интерактивно и то преко посебног терминала који је истовремено и улазни и излазни уређај за корисника. Подела времена је посебан облик мултипрограмирања, где сваком терминалу припада додељено процесорско време. После истека временског квантума тј. додељене количине процесорског времена, процесор се додељује другом терминалу. Уколико је терминал блокиран због чекања на улазно - излазне операције, процесор се и пре истека квантума додељује другом терминалу.[2]

Код треће генерације рачунара посебно треба истаћи појаву два оперативна система:

• MULTICS и
• UNIX.[2]

Пројекат MULTICS (MULTIplexed Information and Computing Service) неуспела је идеја компанија MIT, Bell Labs и General Electric да се направи моћан рачунар и оперативни систем који ће бити у стању да ради са великим бројем терминала. Основна идеја је преузета из модела дистрибуције електричне енергије - у том моделу, довољно је да сваки корисник који пожели да употреби неки електрични апарат, тај исти апарат само прикључи на електричну мрежу. Сличан модел су покушали да направе и са рачунарима: идеја је да у једном раду постоји моћан централни рачунар, а да код куће грађани имају терминале којима преко модема приступају главном рачунару. Овај модел се може сматрати претечом рачунарских мрежа и интернета. Други оперативни систем, UNIX, упрошћена је варијанта MULTICS система, која је доживела практичну реализацију и експанзију до данашњих дана. Ken Thompson, један од научника и програмера компаније Bell Labs, који је радио на развоју пројекта MULTICS, написао је за рачунар PDP-7(Programmed Data Processor) мини верзију MULTICS система. После тога је настао UNIX (UNI = један, X = CS = Computing Service).[2]

Рачунари треће генерације звали су се мини- рачунари: први рачунар је DEC - ов (Digital Equipment Corporation) PDP-1(Programmed Data Processor), до тада најмањи и најјефтинији рачунар. Коштао је тада " само " 120.000 долара. У трећој генерацији рачунарских система, због појаве мултипрограмирања и пораста брзина, величине меморије и броја улазних - излазних јединица, још више контролно - управљачких функција пребацује се са човека на рачунар. Дакле, човек дефинитивно губи могућност контроле интерне ситуације у рачунарском систему и управљања њоме и све што је могуће пребацује се на рачунарски систем тј. на поједине системске програме. Скуп свих тих програма се назива једним именом: оперативни систем. Програмер се ослобађа низа сложених рутинских послова и пружа му се могућност већег ангажовања на креативном делу посла. Поред контролно - управљачких програма у рачунарима треће генерације развијен је и читав низ услужних програма, чији је задатак да олакшају и поједноставе даљу употребу рачунарских система. Због тога, према намени, софтвер можемо поделити на системски и кориснички (апликативни).[2]

У четвртој генерацији рачунара (1980-1990), први пут се појављују персонални рачунари. Развој персоналних рачунара започиње појавом LSI чипова (large scale integration), тј. чипова високог степена интеграције. Рачунари су били довољно јефтини, тако да су их могли приуштити и више одсека исте фирме или универзитета, док су персонални рачунари постали довољно јефтини да их могу имати и појединци. Познатији персонални рачунари су Spectrum, Commodore, Atari, затим IBM PC, Apple Macintosh итд. У прве оперативне системе за персоналне рачунаре спадају MS-DOS i UNIX.[2]

Паралелно с развијањем корисничког софтвера, развија се и кориснички интерфејс програма тј. корисничко окружење. На тај начин се особама које рачунарски систем и саме програме не познају детаљно, омогућава да те програме успешно користе. Поред класичних оперативних система јављају се и две нове врсте, а то су мрежни оперативни системи и дистрибуирани оперативни системи.[3]

Мрежне оперативне системе карактеришу рачунари повезани у мрежу. Ови рачунари задржавају релативно висок степен аутономије - сваки рачунар има свој оперативни систем – а у могућности су да међусобно размењују податке помоћу одговарајућих протокола. Оперативни системи могу бити различити, потребан је само заједнички протокол тј. заједнички језик за комуникацију. Корисник једног рачунара може се пријавити на други, преузети неке датотеке итд. Корисник зна да није сам у мрежи тј. свестан је различитих рачунара с којима комуницира преко мреже.[3]

Дистрибуирани оперативни системи су много озбиљнија варијанта у мрежном окружењу, зато што осим дељења и миграције датотека и штампача омогућавају и дељење процеса тј. програма. Корисници овај систем виде као једнопроцесорски систем, али се, у ствари, ради о оперативном систему намењеном за рад са више процесора који су флексибилно повезани преко мреже. То знаци да постоји више рачунара повезаних у мрежу, али само један

оперативни систем, управља свим ресурсима у мрежи. У правом дистрибуираном систему, корисник не треба да води рачуна о томе где су смештене његове датотеке или где се извршава његов програм - то је посао дистрибуираног оперативног система. Дистрибуирани оперативни систем се, дакле, понаша као јединствена целина. Корисник не мора знати да је умрежен с другим рачунарима - он цео систем види као један рачунар.[3]

3. Архитектура рачунарског система

Архитектура рачунара се односи на целокупну структуру која чини рачунар употребљивим, односно на микропроцесор. Микропроцесори су електронски склопови који функционишу као централно - процесорски - скуп (CPU - Central Processing Unit). Један микропроцесор у свом склопу садржи веома много интегрисаних кола. Интегрисана кола, позната као и чипови, представљају комплексне електронске склопове који се састоје од компоненти поређаних на слој танког, равног комада материјала званог полу - проводник.[10]

Принципи којима подлежу сви рачунари су исти. У основи, сви они узимају сигнале у облику нула (0) и јединица (1), манипулишу њима сагласно неком скупу инструкција и производе излазе, опет у облику нула и јединица.[10]

Процесор ради помоћу реаговања на улаз од више 0 и 1 на одређене начине и враћања излаза заснованог на одлуци. Сама одлука се дешава у електронским склоповима који се зову логичка кола (од којих свако захтева најмање један транзистор), чији су улази и излази различито уређени помоћу различитих операција. Чињеница да данашњи процесори садрже милионе транзистора указује на то колико је сложен такав логички систем. Логичка кола у процесору раде заједно на стварању одлука користећи Боолеову логику.[10]

Ток електрицитета кроз свако коло се контролише помоћу транзистора у том колу. Међутим, транзистори нису појединачне и дискретне јединице. Уместо тога, њихов велики број се производи од једног комада силицијума (или неког другог полу - проводничког материјала) и међусобно повезује помоћу металних проводника или неког другог спољашњег материјала. Овакве јединице се зову интегрирана кола (IC) и њихов развој је, у основи, учинио остваривом сложеност микропроцесора. Интеграција кола се није зауставила на првим резултатима. Баш као што су прва интегрирана кола повезала више транзистора, тако су се касније повезивала и вишеструка интегрирана кола, у процесу који је познат као висок степен интеграције (Large Scale Integration - LSI); на крају су и овакви скупови интегрираних кола били повезивани, у процесу који се зове веома висок степен интеграције (Very Large Scale Integration - VLSI).[10]

Савремени микропроцесори садрже на десетине милиона микроскопски малих транзистора. Употребљени у комбинацији са отпорницима, кондензаторима и диодама, они чине логичка кола. Логичка кола граде интегрирана кола, а интегрирана кола - електронске системе тј. микропроцесоре.[10]

Микропроцесори се класификују према полуводичкој технологији израде на:
- TTL – транзистор- tranzistor logic
- CMOS - complementary-metal-oxide semiconductor и
- ECL - emitter-coupled logic

затим, по величини формата података који обрађују на:
4 – битне
8 – битне
16 – битне
32 – битне
64 – битне
и најзад по њиховом инструкцијском сету на:
CISC - complex-instruction-set computer ili

RISC - reduced-instruction-set computer.[10]

TTL технологија је у најчешћој употреби, док је предност CMOS технологије у погледу преносних рачунара и других уређаја који се напајају из батерија, због изражено умањене потрошње електричне енергије. ECL је у употреби тамо где се захтева већа брзина, а већа потрошња електричне енергије није пресудна . Иако су 4 - битни јефтини модели микропроцесора, добри су једино за једноставне контроле апликације, јер уопштено што је формат података већи - већа је брзина рада, али и цена израде.[10]

Микропроцесори, се пореде са људским мозгом; њихову операцију са контролним таблама итд. Међутим, првобитна сврха микропроцесора, се сводила на контролу меморије. Микропроцесори су дизајнирани са тим циљем у почетку, а то је оно што и дан данас раде![10]

4. CISC и RISC архитектура

Током година на тржишту процесора доминирале су две компаније: Intelov Pentium i Motorolin PowerPC. Ови процесори су такође добри примери две конкурентске архитектуре процесора, CISC и RISC процесори. Класификација процесора у ове две категорије заснована је на скупу инструкција које процесор може извршити; овај скуп називамо инструкцијски скуп.[4]

CISC (*complex instruction set computer*) је традиционална архитектура у којој процесор користи веома бројан скуп сложених инструкција. Инструкција оваквог процесора може бити различите дужине и користити различите начине адресирања. Овај тренд развоја процесора је дуги низ година био запажен међу произвођачима. Међутим, градња оваквих процесора показивала је одређене недостатке (склопови за декодирање постају јако сложени, јавља се низ других

техничких проблема, итд). Већ 1974. IBM-ови стручњаци приступају развоју другачије филозофије градње процесора, покушавајући смањити број инструкција које процесор може извршити. Као резултат тога средином 1980-тих почиње бити доминантан тренд градње процесора који могу извршити само веома мали, ограничен скуп инструкција, сасвим супротно тренду који карактерише CISC процесоре.[4]

RISC (*reduced instruction set computer*) процесори имају инструкцијску реч константне дужине, те задржавају само оне инструкције које могу бити извршене само у једном (или мање од једног) тактном циклусу. Једна од предности RISC процесора је да они могу инструкцију извршити веома брзо, јер је она јако једноставна. Друга, много важнија предност је, да RISC процесори захтевају мање транзистора што их чини јефтинијим за дизајнирање и производњу. Међу експертима и даље постоји низ контраверзи у вези са RISC архитектуром процесора. Једни сматрају да су ово процесори будућности, с друге стране, скептици сматрају да једноставнији хардвер плаћају комплекснијим и скупљим софтвером, јер RISC процесори захтевају сложене компајлере. У пракси, ове две архитектуре постају све близе. Многи данашњи RISC процесори подржавају исто онолико инструкција колико је то био случај за CISC процесоре, и обратно, данашњи CISC процесори користе различите технике раније типичне за RISC архитектуру. Чак и мултинационална компанија међу CISC произвођачима, Intel, употребио је RISC технике у свом чипу 486, поготову у својој фамилији процесора Pentium.[4]

Карактеристике понашања тадашњих CISC процесора и програма који су радили на њима су се заснивале на наредбама, референсирању и позиву процедура. Наредбе које су се најчешће јављале у програмима су биле наредбе доделе, условна наредба и позив потпрограма. Такође, пронађено је да су позив и повратак из процедура операције које захтевају највише времена у типичним програмима писаним на вишим програмским језицима. Реферисање на операнде се обавља као реферисање на локалне скаларне вредности. На основу тога је

закључено да је нужна оптимизација процеса записивања и приступа локалним променљивим. При позиву процедура, око 98% процедура је преносило мање од 6 аргумената, а чак код 92% процедура су ти аргументи били локални. Ови резултати говоре да број речи потребан при позиву процедура није велики, као и да треба обратити пажњу на приступ аргументима, чиме су потврђени ранији резултати. Из резултата студија су следила три начина за побољшање перформанси:

1. Повећање броја регистара. У анализираним програмима постоји јако велики проценат наредби додељивања и померања података. Ова карактерстика заједно са скаларним типом података сугерисала је да се перформансе могу побољшати смањењем реферисања меморије и уместо тога, повећаним реферисањем регистара. Како је највећи број таквих референци локалан, један од практичних прилаза био је увећање броја регистара.[4]

2. Побољшање дизајна преклапања инструкција. Обзиром на високи проценат условних скокова и позива процедура уобичајени начин имплементације преклапања није ефикасан. Последица условних скокова и позива процедура је постојање великог броја инструкција које су дохваћене али се никада не извршавају.[4]

3. Смањен број основних инструкција. Са смањеним бројем основних инструкција једноставније је конструисати микропроцесор (који је по дефиницији на једном једином чипу). Мање времена се троши на препознавање инструкција и инструкције су брже јер се извршавају у једном циклусу.[4]

Најпознатији произвођачи RISC чипова су фирме Motorola (88000, ..., PowerPC), Silicon Grpahics (MIPS R1000, R3000, R4000, ..., R12000), Digital (Alpha), Hewlett Packard (PA-RISC 8200,...,8600), Sun Microsystems (Micro SPARC и ULTRA SPARC) и IBM (RS/6000 i PowerPC). Особине RISC процесора које су заједничке без обзира на произвођача:

1. Извршавање (бар) једне машинске инструкције за један машински циклус. Овим се смањује или елиминише потреба за микрокодом и комплетна машинска инструкција може да буде хардверски кодирана. Таква инструкција се извршава брже од одговарајућих инструкција CISC процесора јер нема потребе за вршењем микропрограмске контроле.[4]

2. Највећи број машинских операција је типа регистар-у-регистар што резултује упрошћеном управљачком јединицом. Такође, оваква архитектура омогућује оптимизацију употребе регистара тако да аргумент коме се често приступа остаје у брзој меморији од које су направљени регистри.[4]

3. Употреба релативно малог броја начина адресирања. Највећи број инструкција RISC процесора користи регистарско адресирање. Поред њега могу да се јаве и други начини адресирања као нпр. адресирање са базним регистром и удаљењем, и / или релативно адресирање у односу на програмски бројач. Остали комплекснији начини адресирања, се реализују софтверски. Ова особина такође има утицај на једноставност конструкције управљачке јединице чиме се повећава брзина рада.[4]

4. Употреба једноставних формата инструкција. У општем случају се користи само неколико различитих формата инструкција. Инструкције су фиксне дужине и обично су поравнате на границу речи, што значи да не прелазе границе страница. Поља у инструкцијама (нпр. код операције) су такође фиксне дужине што омогућује истовремено декодирање операционог кода и приступ операнду инструкције. Такође, мали број формата поједностављује управљачку јединицу.[4]

Предности RISC процесора у односу на процесоре изведене у CISC технологији се могу поделити у две групе:

1. Једноставнија конструкција. Због мањег броја инструкција и једноставније

структуре, потребно време за дизајнирање и увођење таквог процесора у комерцијалну употребу је знатно краће.[4]

2. Боље перформансе. RISC чипови поседују знатно боље перформансе од CISC чипова који раде на истим брзинама. За RISC микропроцесоре је једноставније дефинисати преводиоце који формирају много оптималнији код него за CISC микропроцесоре. Велики број инструкција које генеришу преводиоци је релативно једноставан. Управљачка јединица може да се направи да за овакве инструкције користи врло мало микрокодирања, тако да се оне извршавају брже него на одговарајућим CISC процесорима. [4]

4.1. RISC – архитектура процесора

RISC (reduced instruction set computer) је микропроцесор који је дизајниран да користи мањи број типова инструкција па због тога може да ради на већим брзинама (може да обавља више MIPS - милиона инструкција у секунди). Пошто сваки инструкцијски тип који рачунар треба да реализује захтева додатне електронске склопове, самим тим већа листа инструкција процесора захтева и сложенији процесор који због те своје сложености и великог броја инструкција губи на брзини.[5]

John Cocke из IBM – овог истраживачког центра у Yorktown-у, New York, први је осмислио концепт RISC чак давне 1974. јер је доказао да 20 % инструкција обавља 80 % " посла " у рачунару. Први рачунар који је био усавршен и који је имао користи од овог открића је био IBM - ов PC/XT из 1980. године. Касније IBM - RISC систем/6000 постао је један од најпознатијих RISC рачунара. Термин RISC је креирао David Petersen предавач на Берклију, универзитет у Калифорнији. Овај концепт је коришћен у SPARC микропроцесорима Sun мицросистем компаније и довео је до оснивања оног што је данас познато као MIPS (Million Instructions per

Second) Technologies. RISC технологију такође користе и DEC – ови (Digital Equipment Corporation) Alpha микрочипови.[5]

RISC концепт је тако довео до највише осмишљенијег дизајна архитектуре микропроцесора. Међу главним разматрањима приликом дизајна микропроцесора је и колико се добро инструкција може мапирати на брзину такта микропроцесора (у идеалном случају, се извршава у једном такт периоду); колико ће дата архитектура бити поједностављена и колико ће послова микропроцесор моћи да обави сам са тим редукованим сетом инструкција без неке посебне софтверске помоћи. Поред побољшања перформанси, неке предности RISC су:

• Нови микропроцесор, се може произвести и тестирати много брже ако му је дизајн мање компликован .
• Програмерима је знатно олакшан посао ако користе мањи инструкцијски сет.
• Једноставност RISC омогућава више слободе приликом коришћења простора на кристалу процесора приликом његовог дизајна и израде.
• Компајлери језика вишег нивоа производе ефикаснији и бољи код него пре.[5]

За сваку инструкцију RISC система најчешће је потребно само један такт да се изврши, па су зато RISC системи способни за паралелну обраду инструкција у процесу који се назива цевовод. Процесор обрађује тад више од једне инструкције истовремено. Због тога је пропусна моћ RISC система знатно већа и зато су они много бржи од CISC система. Лоше дизајниран инструкцијски сет може довести до честих застоја гасовода архитектуре. Неки од најчешћих проблема о којима, се мора водити рачуна при дизајну инструкцијског сета су:

• Инструкције би требале бити исте дужине (извршавају се у једном такт периоду) јер у противном долази до вишеструког референцирања приликом прибављања инструкције из меморије .
• Што мање инструкција које приступају главној меморији јер је главна меморија релативно спора (боље је користити регистре у самом процесору)

• Што мање инстукција које раде са једним регистром (нпр. " додати 5 регистру A2" прво очитава регистар A2 , затим додаје вредност 5 тој вредности и онда поново смешта резултат у исти регистар који може бити " заузет " од претходне операције читања, што узрокује застојем док се не добије приступ регистру).
• Што мање инструкција које користе једноструке показиваче као што је регистар стања . Ако једна инструкција поставља стања у регистру стања а идућа инструкција покуша да очита те битове, тада ће та друга инструкција можда бити заустављена док се операција уписа прве инструкције не заврши.[5]

Једна од најпознатијих метода паралелне обраде код RISC процесора је суперскаларна архитектура. У суперскаларној машини, процесор управља са више инструкцијских цевовода да би извршио више инструкција истовремено за време трајања једног такт периода (или да би извршио једну инструкцију за коју треба више такт периода). Ово је решено тако да се различити гасоводи снабдевају кроз одређен број извршних јединица унутар процесора. Да би се успешно имплементирала суперскаларна архитектура потребно је да је механизам за прибављање и извршавање инструкција довољно добар јер се у супротном може десити да дође до загушења цевовода која би онда довела до честих застоја у раду извршних јединица. Застоји се могу такође десити кад извршна јединица покуша да изврши задатак који зависи од неке инструкције која још није до краја завршена. Зато је веома важно да се пажљиво управља редоследом извршавања инструкција.[5]

Са друге стране CISC системи да би убрзали рад покушавају да смање број инструкција које програм мора позвати. Због тога имају велики број инструкција микрокода које покривају широки спектар задатака. Пошто једна инструкција микрокода, када се преведе унутар процесора, може постати задатак који се решава у више корака. Као резултат се инструкције извршавају најчешће у више такт периода што знатно успорава рад процесора. Примери CISC процесора су 680x0, x86 и VAX. RISC процесори су PowerPC, MIPS, SPARC и Alpha. Intel свој Pentium II назива CRISC процесором (Complex Reduced Instructions Set Computer)

јер је код његовог дизајна коришћена хибридна архитектура која искоришћава предности CISC и RISC архитектура.[5]

Двадесетак година индустријске производње процесора довели су до захтева за ефикаснијим радом процесора. Тај захтев пре свега подразумева минималан број наредби кодирања и декодирања у процесору. На бази тог захтева, развијена су хардверска решења процесора за све операције. То је довело до значајног упрошћавања многих процедура процесора, а створене су могућности за вишепроцесорски рад. [5]

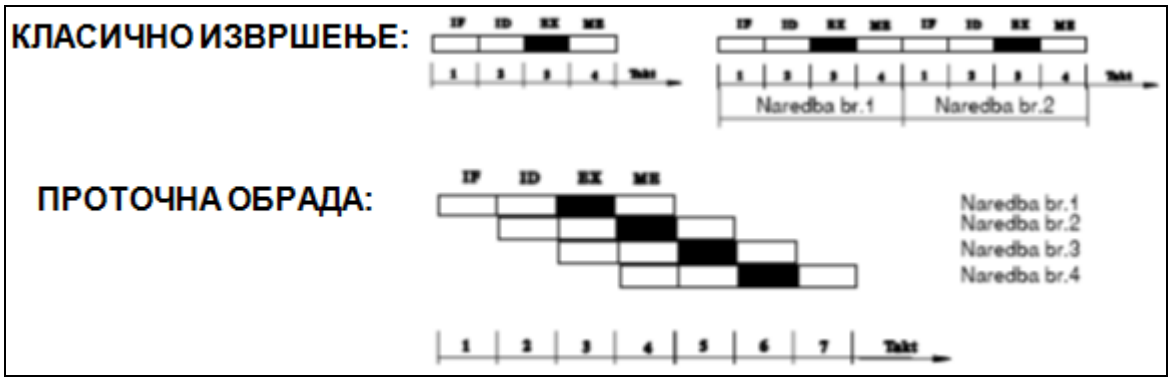
Потреба за повишењем радне поузданости утицала је да се уведе паралелизам у издавању наредби (логичка анализа). Овако пројектовани процесори представљају нову генерацију, а њихова архитектура позната је под називом Reduced Instruction Set Computing (RISC). У хардверском смислу RISC процесори имају обележја суперрачунара, јер интегришу векторски процесор, вишепроцесорску магистралу, хардверску подршку другостепеном кеш меморијом.[5]

Први Intel - ов RISC процесор 80860 направљен је 1989. Четири оваква процесора при радном такту од 50 MHz имају снагу од 400 MFLOPS-a. Овај процесор садржи векторски копроцесор (Floating Point копроцесор), 3Д графички процесор и друге микропроцесорске компоненте опште намене. Магистрала овог процесора је 64 - битна, док је магистрала између двеју унутрашњих кеш меморија 128 - битна. Кеш меморије капацитета 4 KB и 8 KB, намењене су за смештај наредби и смештај података. Овај процесор користе различити произвођачи као што су IBM, Sun, HP за своје графичке радне станице. По угледу на овај процесор настали су и други процесори као AMD 29050, Fujica SPARC итд. Обављање једне наредбе има четири фазе:

• IF - прихватање информације из меморије (Instruction Fetch) ,
• ID - декодирање информације (Instruction Decode) ,

• EX - извршење (Execute) и
• ME - обраћање меморији за смештај резултата извршења.[5]

Код класичних рачунара наредбе се одвијају једна за другом - секвенцијално. Код RISC архитектуре наредбе се могу извршити паралелно, што је познато под појмом проточне обраде података (цевовод). Разлика је у времену трајања обраде: док је код класичних процесора био потребан по један такт за сваку фазу реализације, сада је за читаву наредбу (за све фазе) потребан један радни такт. Ово наравно може да се постигне само ако процесор интегрише више идентичних извршних јединица. То показује слика 1. На бази RISC архитектуре су развијене и друге архитектуре процесора као што су суперскаларна, суперпипелине, VLIW цевовода , вектор цевовода и друге. Најсавременији RISC процесори, се израђују на бази галијума и арсена и дају веће радне брзине. Пробни узорци раде на радном такту од 2500 MHz и лако постижу пропусност од 2000 MIPS-a (Million Instructions per Second). У погледу величина, RISC процесори имају најмање дужине путања логичких кола. Ови процесори се користе за графичке радне станице (IBM RS/6000).[5]



Слика 1. Поређење класичне и проточне обраде [5]

5. ARC

Простудираћемо модел архитектуре који је заснован на комерцијалном Scalable Processor Architecture (SPARC) процесору који је развијен у Sun микросистемима средином осамдесетих. SPARC је постао популарна архитектура од свог увођења, делом због своје отворене природе. Пуна дефиниција SPARC архитектуре је спремно представљена као доступна јавности (SPARC, 1992). Погледаћемо само подскуп SPARC_а који називамо A RISC Computer (ARC). " RISC " је још један акроним, за редуковани инструкцијски сет компјутера. ARC има већину важних одлика SPARC архитектуре, али без неких комплекснијих одлика које су присутне у комерцијалном процесору.

5.1. ARC меморија

ARC је 32-битна машина са меморијом бајт адреса. Може манипулисати са 32-бит типовима података, али су сви подаци похрањени у меморији као бајтови и адреса 32-бит речи је адреса њеног бајта који има најнижу адресу. ARC има простор 32-бит адреса, у којем је наш пример архитектуре подељен на издвојене регије за употребу од стране системског оперирајућег кода, кода корисничког програма, системског асемблера (system stack) (коришћеног за похрањивање привремених података), і input і autput, I/O.[1]

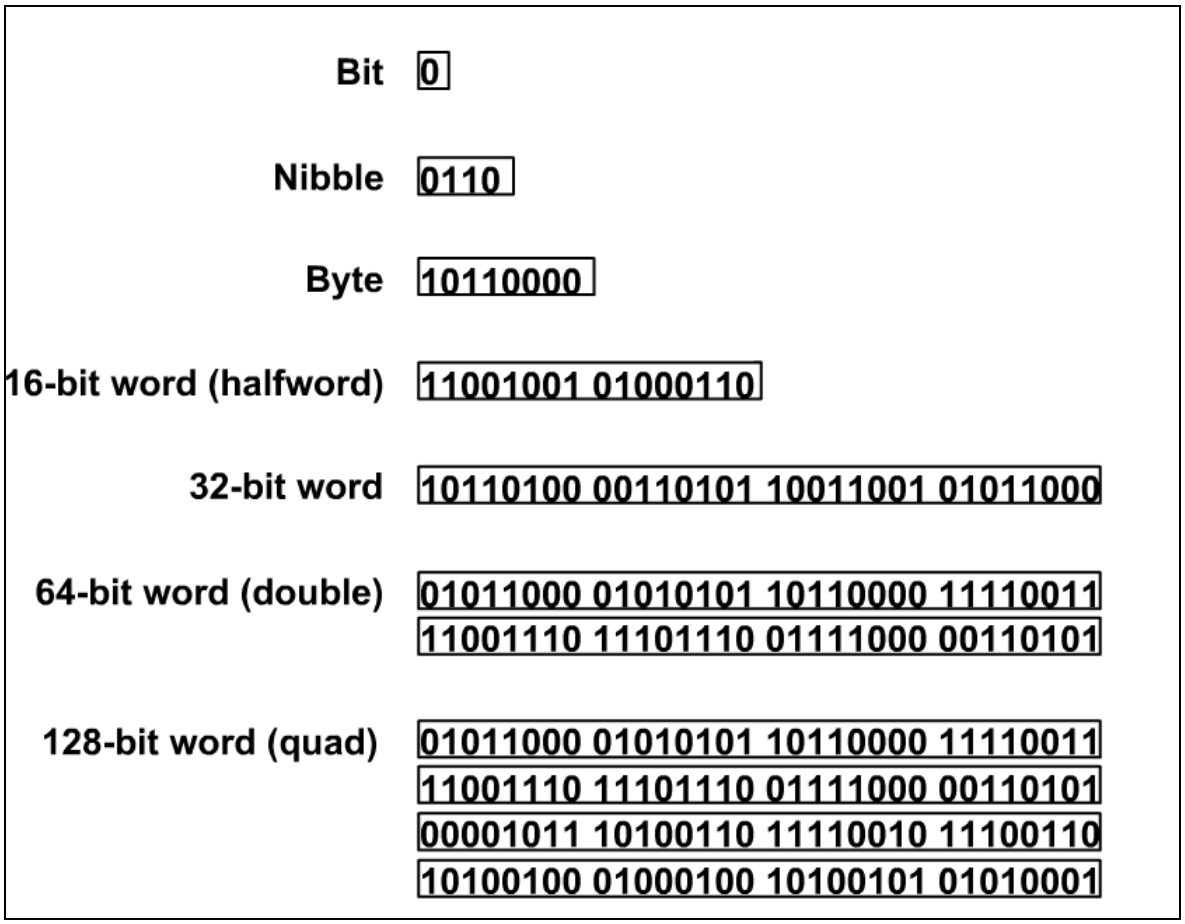
ARC има неколико типова података (бајт, полуреч, цео број, итд.) али сад ћемо узети у обзир само 32-бит цео број тип података. Сваки цели број је похрањен у меморији као колекција четири бајта. ARC је big-endian архитектура, тако да је бајт највишег реда похрањен на најнижој адреси. Највећа могућа бајт адреса у ARC -у је 2^{32}-1, тако да је адреса од највише речи у меморијској мапи три бајта нижа од ове, или 2^{32}-4.[1]

Компјутерска меморија се састоји од колекције регистара који су нумерисани (адресирани) и сваки садржи један бајт. Бајт је збирка од осам бита (понекад се она у рачунарској комуникацији назива октет). Сваки регистар има адресу и то се назива меморијска локација. Nibble или Nybble се односи на прикупљање четири везана бита. Значење израза "бит", "бајт" и "nib-ble" је обично договорено без обзира на специфичности архитектуре, али значење речи зависи од конкретног процесора. Типичне величине низова су 16, 32, 64, и 128 бита, док је 32-битна величина данас уобичајена форма за обичне рачунаре, а употреба 64-битне величине постаје све популарнија. Поређење ових типова података је приказан на слици 2.[1]

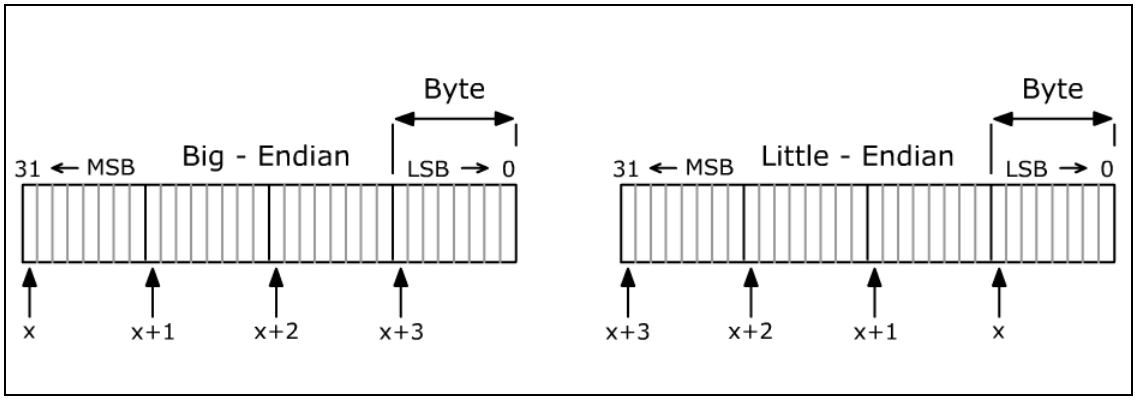
У бајт-адресабилне машине, најмањи објекат који може бити наведен у меморији је бајт. Међутим, обично постоје инструкције које читају и пишу мулти-бајт речи. Мулти-бајт речи се чувају као низ бајтова, адресиране по бајт речи која има најнижу адресу. Већина машина данас имају инструкције које могу да приступе бајтовима, полу-речима, речима, и дуплим речима.[1]

Када се користе мулти-бајт речи, постоје две опције о редоследу чувања бајтова у меморији: најзначајнији бајт на најнижој адреси, назива се big-endian, или најмање значајан бајт чуван на најнижој адреси, назива се little-endian. Термин " endian " потиче од питања да ли јаја треба да буду сломљена на великом или малом крају, што је изазвало рат од препирки политичара у Гуливеровим путовањима Jonathan Swift's. Примери big-endian и little-endian формата за 4-бајта, 32-битне величине су приказани на слици 3.[1]

Меморијске локације су линеарно распоређене у редоследу као што је приказано на слици 3. Свака од нумерисаних локација одговара одређеној чуваној речи (овде се реч састоји од четири бајта). Јединствени број који идентификује сваку реч назива се његовом адресом. Пошто се адресе броје у низу почевши од нуле, највиша адреса је за један мање од величине меморије. Највећа адреса за 2^{32} бајта меморије је 2^{32}-1.Најнижа адреса је 0. [1]



Слика 2. Уобичајене величине за типове података [1]



Слика 3. Big-endian и little-endian формата [1]

5.2. ARC формати података

ARC подржава 12 различитих формата података. Формати података су сврстани у три групе: означени цео број, неозначени цео број и плутајућа тачка (floating point). У оквиру ових типова, дозвољене ширине формата су бајт (осам битова), полуреч (16 бита), реч / самореч (32 бита), означене речи (које се повлаче) (32 бита у којој два најмање значајна бита чине ознаку и најзначајнија 30 бита која формирају вредност), дупла реч (64 бита), и quad реч (128 бита).[6]

ARC не прави разлику између означених и неозначених целих бројева. Оба су похрањена и њиме се манипулише као са двојним комплементним целим бројем. Њихово тумачење је оно које варира. У једном случају, подскуп огранка инструкција претпоставља да су вредности које се упоређују означени цели бројеви, док други подскуп треба да буду неозначени. Слично томе, C бит показује преливање неозначених целих бројева док V бит означава преливање означених целих бројева.[6]

Tagged реч користи два најмање значајна бита да би указивали на преливање, у којој се покушава да се сачува вредност која је већа од 30 бита, у издвојених 30 бита 32 -битне речи. Tagged аритметичке операције користе се у стандардизованим језицима са динамичким подацима, као што су Lisp i Smalltalk. У свакој општој форми, 1 у било којем биту таг поља указује ситуацију преливања за ту реч. Ознаке се могу користити да би се обезбедили одговарајући услови поравнања (речи које почињу на 4 - битним границама, quad речи на 8 - битним границама, итд), нарочито за усмериваче. Формати плутајуће тачке (floating point) су прилагођени за IEEE754 (Institute of Electrical and Electronics Engineers) стандард.[6]

5.3. ARC инструкцијски формати

Инструкцијски формат одређује како су разна поља битова инструкције формирани од стране асемблера и како су протумачени од стране ARC контролне јединице. ARC архитектура има само неколико инструкцијских формата. Пет формата су : SETHI, Branch, Call, Arithmetic и Memory, како је приказано на слици 4.[6]

Свака инструкција има мнемоничку форму као што је " ld " и " opcod ". Одређени инструкцијски формат може имати више од једног opcod поља, које колективно идентификују инструкцију у једној од разних форми.[6]

Два лева бита сваке инструкције из op поља (opcode) су та која идентификују формат. SETHI и Branch формати садрже 00 у op пољу, те се заједно могу посматрати као SETHI/Branch формат. Стварни SETHI или Branch формат је одређен узорком бита у op2 opcode пољу (010 = Branch; = SETHI). Бит 29 у Branch формату увек садржи нулу. Пет - битно rd поље идентификује циљани регистар за SETHI операцију.[6]

Cond поље идентификује тип огранке, на основу битова кодова стања (condition code bit) (n, z, v, i c) u PSR (Power Soft Report), као што је показано на слици 4. Резултат извршења инструкције у којој mnemonic завршава са " cc " поставља битове кодова стања као што су y n = 1 ако је резултат операције негативан; z = 1 ако је резултат нула; v = 1 ако операција узрокује преливање (overflow); и c = 1 ако операција произведе пренос (carry).[6]

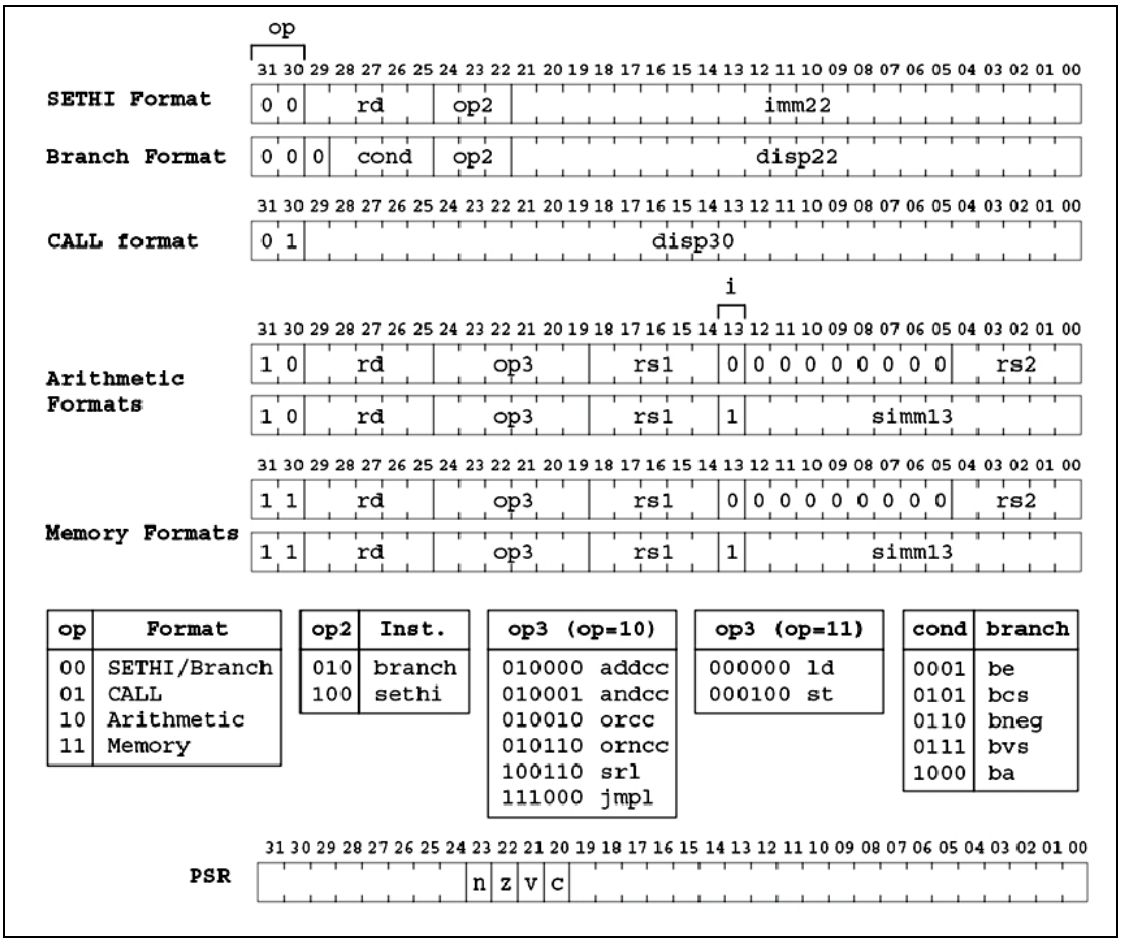
Инструкције које не завршавају на " cc " не утичу на кодове стања. Imm22 и disp22 поља садрже 22 - битну константу која се користи као операнд за SETHI формат (за imm22) или за израчунавање промењеног распореда (displacement) адреса огранака (за disp22).[6]

CALL формат садржи само два поља; op поље које садржи узорак бита 01, и disp30 поље, које садржи 30 - битни промењени распоред који се користи у израчунавању адреса позване рутине.[6]

Аритметички (op = 10) и меморијски (op = 11) формати користе rd поља било да би идентификовали изворни регистар за st, или одредишни регистар за преостале инструкције. Rs1 поље идентификује први изворни регистар irs2 поље идентификује други изворни регистар. Op3 opcode поље идентификује инструкцију према op3 табелама приказаним на слици 4.[6]

Simm13 поље је 13 - битна непосредна вредност која је знаковно проширена на 32 бита за други извор кад је и (immediate) поље 1. Значење " знаковно проширен " је у томе да је леви бит simm13 поља (знаковни бит) копиран на лево у преостале битове који сачињавају 32 - битни цео број, пре његовог додавања на rs1 у овом случају. Ово осигурава да други двојчани комплементни негативни број остане негативан (и други позититивни комплементни број остане позитиван). На пример, (-13) 10 = (1111111110011) 2 и након знаковног проширења на 32 – битни цео број, ми онда имамо (11111111111111111111111111110011) 2 који је још еквивалентан са (-13) 10.[6]

Аритметичким инструкцијама су потребна два изворна операнда и одредишни операнд, укупно три операнда. Меморијским инструкцијама су потребна само два операнда, један за адресу и један за податке. Међутим, преостали изворни операнд се такође користи за адресу. Операнди у rs1 i rs2 пољу и у simm13 пољу су додати да би одржавали адресу.[6]



Слика 4. Инструкцијски и PSR формат за ARC [6]

5.4. ARC инструкцијски сет

Инструкцијски сет је колекција инструкција које процес може извршити и по свом ефекту дефинише процесор. Инструкцијски сетови за сваки тип процесора су у потпуности различити једни од других. Разликују се по величини инструкција, врсти операција које омогућују, типу операнда и типовима резултата које дају. Преглед CPU-a (central processing unit) - централна процесорска јединица:

- ARC има 32 регистра опште намене.[6]

- Ту је и Processor Status Register (PSR) који садржи информације о стању процесора, укључујући и информације о резултатима аритметичких операција. Аритметичка " застава" у PSR, се назива код стања (condition code). Он одређује да ли је одређена аритметичка операција резултирала нулом (z), негативном вредношћу (n), изводом из 32 – битних аритметичких и логичких јединица - ALU (c), или преливањем - overflow (v). V бит се добије кад су резултати аритметичких операција превелики да би их ALU-е (Arithmetic Logic Unit) могле користити.[6]

- Све инструкције су у величини једне речи (32 бита).[6]

- ARC је load-store машина (уношење - похрањивање), чији је једини доступни приступ меморије ка операцијама (allowable memory access operations) спрема вредност на један од регистара, или похрањује вредност садржану у једном од регистара у меморијској локацији. Све аритметичке операције оперишу на вредностима које се налазе у регистрима и резултати се смештају на регистар. Постоји отприлике 200 инструкција у SPARC инструкцијском сету, на основу којих се ARC инструкцијски сет заснива. Подскуп 15 инструкција је приказан на слици 5. Свака је инструкција представљена са мнемоничком, што је име које представља инструкцију.[6]

Слика 5. Подскуп инструкцијског сета за ARC [6]

5.4.1. Меморијске инструкције

Најважније две инструкције: ld (load) и st (store) преносе реч између главне меморије и једног од ARC регистара. Ово су једине инструкције које могу ући у меморију у ARC.[6]

Sethi инструкција поставља 22 најзначајнија бита регистра са 22 бита константно садржана унутар инструкције. Обично се користи за конструисање арбитрарне 32-бит константе у регистру, у коњукцији са другом инструкцијом која поставља 10 бита ниског реда у регистру.[6]

5.4.2. Логичке и аритметичке инструкције

Andcc, orcc i orncc инструкције изводе бит - по - бит AND, OR i NOR операције, овисно о њиховим операндима. Један од два изворна операнда мора бити у регистру. Други може бити или у регистру, или може бити 13 - бит друга допунска константа која се садржи у инструкцији, која је знаковно проширена на 32 - бита кад је у употреби. Резултат је похрањен у регистру.[6]

За andcc инструкцију, сваки бит разултата је одређен на 1 ако су оба одговарајућа бита оба операнда 1, иначе је одговарајући бит резултата одређен на 0. Orcc операција је допуна orncc-а, тако да је сваки бит резултата 0 ако су иједан од битова одговарајућих операнда 1, иначе је бит резултата одређен на 1. " cc " суфикси одређују да након извођења операције, да се битови кода стања ревизирају да би очитавали резултате операције. z бит је одређен ако резултирајући регистар садржи све нуле, n бит је одређен ако је најзначајнији бит резултирајућег регистра 1, и c и v заставе су разјашњене за ове одређене инструкције.[6]

Shift инструкције шаљу садржаје из једног регистра у други. Srl (shift right logical) инструкција шаље регистар на десно и копира нуле у леве битове. Sra (shift right arithmetic) инструкција (није приказана), шаље оригиналне садржаје регистра на десно смештајући копију MSB-а оригиналног регистра на ново створене слободне битове или бит на десној страни регистра. Ово резултира у знаковном проширивању броја и са тиме чува свој аритметички знак. Addcc инструкција изводи 32 - бит други комплементарни прилог на својим операндима.[6]

5.4.3. Контролне инструкције

Call и jmpl инструкције формирају пар који се користи за позив и враћање из потпрограма, овисно о случају. Јмпл се такође користи ради контроле преноса у други део програма.[6]

Нижих пет инструкција се зову инструкције Условних огранака (conditional branch instructions). Be, bneg, bcs, bvs, и ba инструкције стварају огранак при извршењу програма. Називају се условни јер тестирају један или више битова кода стања у PSR (Processor State Register), па ако битови индицирају стање сусреће се огранак. Користе се у имплементирању конструката високог нивоа као што су goto, if-then-else и do-while. [6]

5.5. Додатак ARC инструкцијама(псеудо-операције)

Као додатак ARC инструкцијама које подржава архитектура, налазе се псеудо- операције (псеудо-опс) које уопште нису опкодови, већ пре инструкције асемблеру да се изводе неке акције у асембли време.[6]

Листа псеудо - опс - а и примери њихове употребе су приказани на слици 6. Можете приметити да за разлику од опкодова процесора, који су специфични за одређену машину, врсте и природа псеудо-опса је специфична за дати асемблер, јер се извршавају од стране самог асемблера.[6]

.equ псеудо-опс даје инструкције асемблеру да изједначи вредност (equate = једначина) или да ознаку повеже са симболом, тако да се симбол може користити кроз програм као да је ознака или веза (string) написана на свом месту. .begin и .end псеудо-опс говоре асемблеру кад да почне и да заврши асемблирање. Било које изјаве које се појаве пре .begin или након .end се

игноришу. Један програм може имати више од једног .begin/.end пара, али ту мора бити .end за сваки .begin и мора постојати бар један .begin. Употреба .begin и .end су корисни у стварању делова програма који су невидљиви за асемблер током "требљења" (debugging).[6]

.org (оригин - порекло) псеудо - оп узрокује асемблирање друге инструкције са претпоставком да ће бити смештена у специфицирану меморијску локацију у време рада (runtime).[6]

.dwb (define word block - дефинирај блок речи) псеудо - оп резервише блок 4 - битних речи, што је типично за приказ. Локацијски бројач (који води евиденцију о томе која се инструкција асемблира од стране асемблера) се помера испред блока у зависности од броја речи специфицираних од стране аргумента на .dwb помноженом са 4.[6]

.global и .extern псеудо - опс баве се са именима променљиве и адреса које су дефинисане у једном модулу асембли кода и користе се у другом.[6]

.global псеудо - оп чини ознаку (label) доступну осталим модулима.[6]

.extern псеудо - оп идентификује ознаку која је коришћена у локалном модулу и дефинисана је у другом модулу (који би требао бити означен са .global у том модулу).[6]

```
.equ      X   .equ   #10          Treat symbol X as (10)16
.begin        .begin              Start assembling
.end          .end                Stop assembling
.org          .org   2048         Change location counter to 2048
.dwb          .dwb   25           Reserve a block of 25 words
.global       .global Y           Y is used in another module
.extern       .extern Z           Z is defined in another module
.macro        .macro M a, b, ...  Define macro M with formal
                                  parameters a, b, ...
.endmacro     .endmacro           End of macro definition
.if           .if <cond>          Assemble if <cond> is true
.endif        .endif              End of .if construct
```

Слика 6. Пример асемблер (assembly language) програма [6]

Процес писања асемблер програма је сличан процесу писања програма високог нивоа, осим што су многи детаљи који су уклоњени у програмима високог нивоа, учињени у експлицитним асемблер програмима. У овом делу посматрађемо пример сабирања два броја.[6]

Програм : Сабирање 2 цела броја

Подразумева писање ARC асемблер програма који сабира бројеве 15 и 9. Једно могуће кодирање је приказано у примеру на слици 7. Програм почиње и завршава са .begin/ .end паром. . org псеудо - оп даје инструкцију асемблеру да почне асемблирање тако да је асемблиран код пребачен у меморију почињући на локацији 2048.[6]

Операнди 15 и 9 су похрањени у променљивама x и y, зависно од случаја.[6]

Могу се само додати бројеви похрањени у регистре у ARC - у (зато што само ld и st могу ући у главну меморију) и тако програм почиње са пребацивањем регистара %r1 i %r2 са x и y.[6]

Addcc инструкција додаје %r1 и %r2 и смешта резултат у %r3. St инструкција онда складишти %r3 у меморијску локацију z. jmpl инструкција са операндима %r15 + 4, %r0 узрокује повратак на следећу инструкцију у позивној рутини, која је оперативни систем ако је ово највиши ниво корисничког програма као што можемо претпоставити да јесте. Променљиве x, y, и z прате програм.[6]

У пракси, SPARC код еквивалентан ARC коду приказан у примеру на слици 7. није у потпуности тачан. ld, st, и jmpl инструкцијама је потребно скоро два циклуса да се комплетирају и пошто SPARC почиње са новом инструкцијом сваки пут кад сат куцне, ове инструкције требају бити праћене инструкцијом која не зависи о њиховим резултатима . Ово подручје (property) лансирања нове инструкције пре него што је предходна комплетирана, назива се pipelining.[6]

```
овај програм сабира два броја

        .begin
        .org 2048
prog1:  ld      [x],    %r1             !Load x into %r1
        ld      [y],    %r2             !Load y into %r2
        addcc   %r1,    %r2,    %r3     !%r3←%r1 + %r2
        st      %r3,    [z]             !Store %r3 into z
        jmpl    %r15 + 4,       %r0     !Return
x:      15
y:      9
z:      0
        .end
```

Слика 7. ARC асемблерски програм сабирања два броја [6]

6. Закључак

ARC је представио прво у индустрији кориснички прилагодљиво 32 - битно RISC процесорско језгро. RISC микропроцесорска језгра могу да се имплементирају на процесор и имају подршку комплетног пакета развојних алата. Данас представља кључ за уграђена решења која комбинују у реалном времену оперативни систем, развојне алате и периферни хардвер и софтвер који омогућава програмерима да боље оптимизују дизајн и перформансе. ARC обезбеђује један извор за велике блокове SoC (system on a chip) и уграђене блокове софтверских процесора, смањење броја добављача, смањење трошкова, смањење ризика и смањује време потребно за тржиште.[7]

ARC International је светски лидер у мале снаге, високих перформанси 32 - битни CPU (central processing unit)подесиви / DSP (Digital signal processing) процесорских језгара, подсистема, real-time оперативног система и развојних алата за пројектовање уграђених система. Корени ARC International PLC (Programmable Logic Controller) датирају из раних 1990-их. ARC је подесиво телескопско језгро и помаже клијентима у развоју следеће генерације дигиталних медија, потрошачких и комуникационих уређаја, што доводи до нижих трошкова.[8]

7. Литература

[1] M.J. Murdocca, V.P. Heuring, *„Principles of computer architecture“,* Prentice Hall International, Inc. pp. 101-125, New Jersey 2000.

[2] Д.Петровић, *„Асемблерски преводилац за 16-битни RISC процесор са вон Нојмановом архитектуром”,* дипломски рад, Електротехнички факултет, Београд 2007.

[3] Texas Instruments *„MSP430x44xx Family User's Guide”,* Vol 3. pp. 41-115, 2006.

[4] М. Стојчев, *„RISC, CISC и DSP процесори“,* I издање, Електронски факултет, Ниш, 1997.

[5] J. Ђорђевић, M. R. Barbacci, B. Hosler, A PMS Level Notation for the Description and Simulation of Digital Systems, The Computer Journal, Vol. 28, No. 4, pp. 357-365, 1985.

[6] http://dl.fit.ba/nastava/ars/poglavlje_10/10.3.2/index.html (приступљено 1.10.2013.)

[7] http://www.design-reuse.com/news/6447/arc-international-configurable-processor-architecture-power-sensitive-embedded-systems.html (приступљено 1.10.2013.)

[8] http://www10.edacafe.com/nbc/articles/view_article.php?section=ICNews&articleid=165541 (приступљено 1.10.2013.)

[9] http://poincare.matf.bg.ac.rs/~cvetana/Istorijat/IIWW/IIWW.html (приступљено 1.10.2013.)

[10] http://forum.bih.net.ba/showthread.php?t=41857&s=17c695887d328790ff70f04dc60b2580 (приступљено 1.10.2013.)