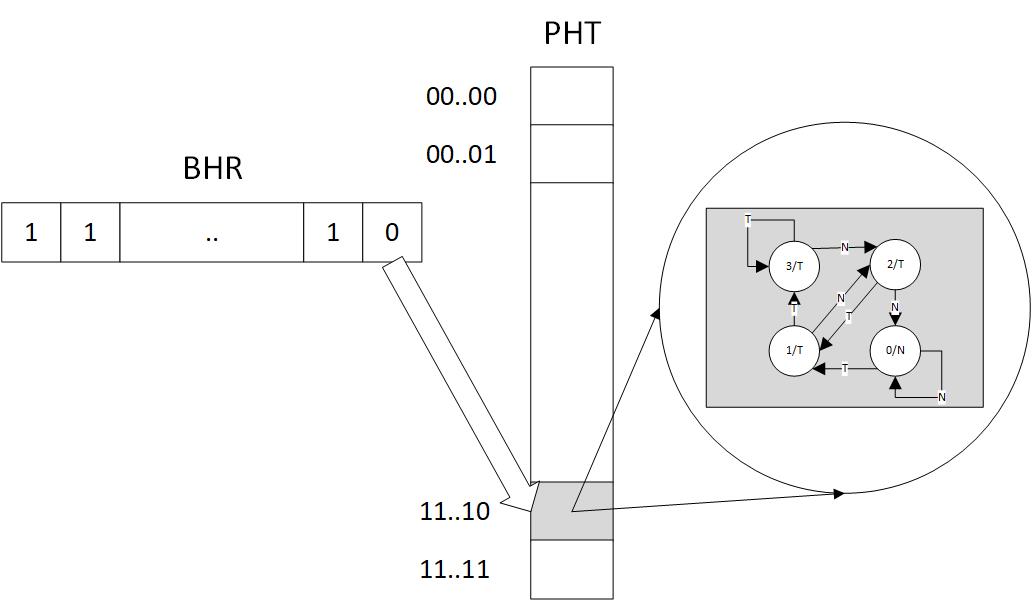
* 1. Постојећи системи за предвиђање скокова

У овом потпоглављу биће представљени неки од система за предвиђање скокова који су доступни у отвореној литератури. Ови системи ће бити разматрани са стране предвиђања исхода скокова, неће се разматрати предвиђање одредишне адресе скока. Сви ови системи спадају у групу динамичких система за предвиђање, јер предвиђање рачунају на основу информација које су сакупили током извршавања. У наставку овог рада равноправно ће се користити и термин предиктор уместо термина систем за предвиђање скокова.

* + 1. Дво-нивовски предиктор

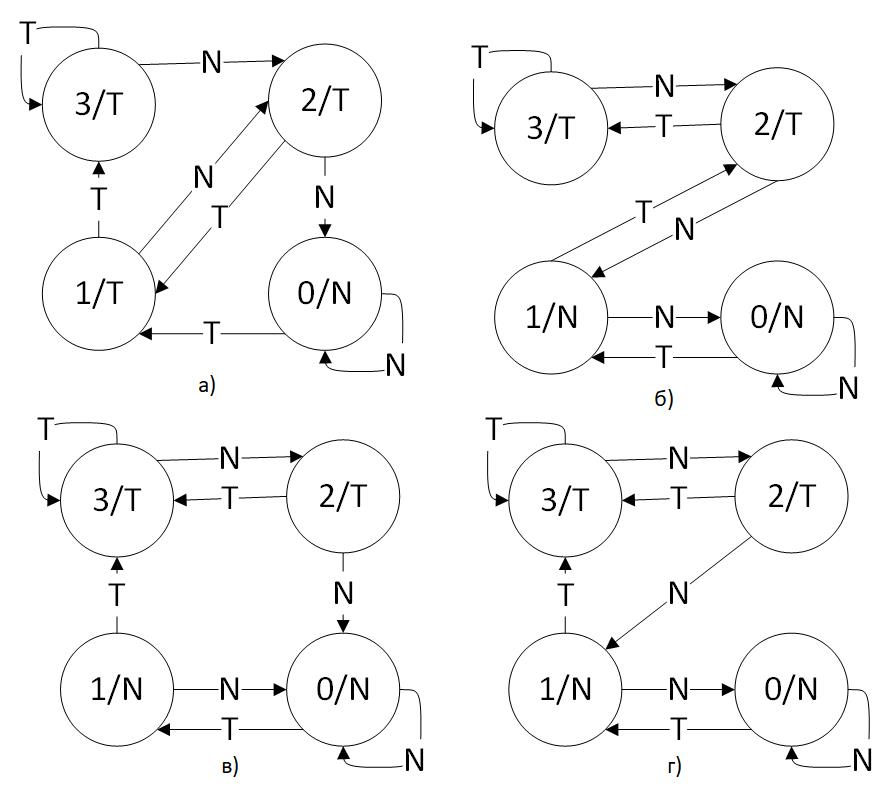
Овај систем који је описан у раду [4] чине два основна дела који се називају *Branch History Register* (BHR) и *History Pattern Table* (PHT). BHRпредставља регистар који служи да памти историју раније извршених скокова и представља први ниво овог предиктора. То је померачки регистар који се након утврђеног исхода скока ажурира тако што се помера у лево за један бит и на место најнижег бита се уписује јединица или нула на основу тога да ли се скок десио или не, респективно. На основу садржаја овог регистра врши се приступање једном од улаза у табели PHT, која представља други ниво овог предиктора*.* Улази у овој табели садрже вредности које представљају стање аутомата на основу кога се одређује предикција. На слици 2.2.1.1. је приказана принципска шема овог система са једним глобалним регистром BHR и једном глобалном табелом PHT.

Поред глобалног BHR овај систем може имати и више BHR [5]. Постоје две варијанте са више BHR. Код прве варијанте један BHR памти историју само једног скока (*per address*), док код друге варијанте један BHR памти историју скокова који припадају истој групи (*per set*). Припадност групи се одређује на основу неких битова из адресе инструкције скока. Поред глобалне PHT овај систем може имати и више PHT. Постоје две варијанте са више PHT. Код прве варијанте једна PHT чува предвиђање (аутомат стања) само једног скока (*per address*), док код друге варијанте једна PHT памти историју скокова који припадају истој групи (*per set*).



Слика 2.2.1.1. Принципска шема дво-нивовског предиктора [4]

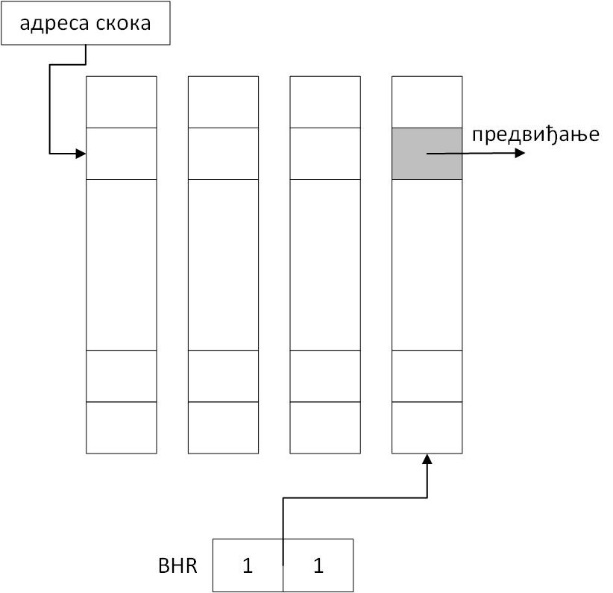
Улаз из табеле PHT који се очита представља стање аутомата на основу кога се добија предвиђање. Аутомати који се користе су аутомати са четири стања, тако да је ширина улаза у табели два бита. На слици су приказани могући прелази између стања аутомата који су аутори рада [4] предложили. На слици 2.2.1.2. су приказана четири аутомата. Сваки аутомат има четири стања. Прелазак између стања се обавља када се утврди стваран исход скока, акције прелаза су означене са *T* или *N* у зависности од тога да ли се скок десио или није, респективно. Стања су нумерисана и означена симболима *Т* или *N* на основу којих се одређује предвиђање да ли ће се скок десити или не, респективно.



Слика 2.2.1.2. Аутомати стања на основу којих се врши предикција [4]

* + 1. Корелациони предиктор

Група аутора је предложила овај систем за предвиђање скокова у раду [6]. Они су приметили да на исход неког скока утичу и исходи инструкција скокова које су се извршиле пре посматраног скока. Њихов систем користи један глобални померачки регистар који памти историју свих скокова, исто као BHR код дво-нивовског предиктора. На основу садржаја тог регистра одабира се једна од табела која представља један део историје исхода скокова. Те табеле служе да дају предвиђање исхода скока. На слици 2.2.2.1. је приказана принципска шема овог система.



Слика 2.2.2.1. Корелациони предиктор са двобитним глобалним регистром [6]

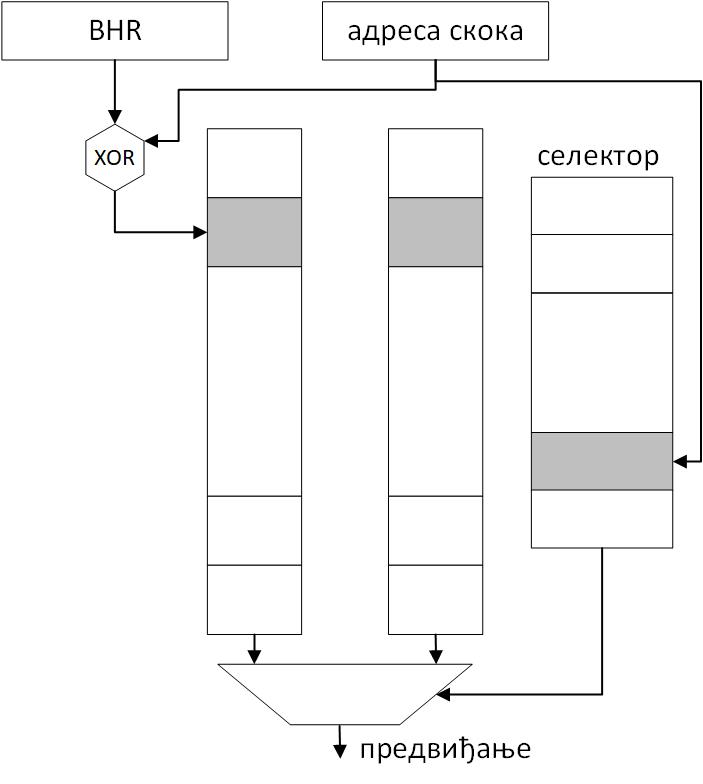
На слици 2.2.2.1. се види да је вредношћу глобалног регистра историје одабрана једна од табела, а да се улаз унутар одабране табеле одабира на основу најнижих битова из адресе скока. Улази у овој табели садрже аутомате стања на основу којих се врши предвиђање. Приликом ажурирања, ажурира се аутомат стања одабраног улаз и глобални регистар историје на основу коначног исхода скока.

Овај систем се означава као (M, N) корелациони предиктор. M представља ширину у битовима глобалног регистра историје, док N представља број бита који се користе за означавање стања аутомата. Величина параметра М може бити произвољна и на основу ње се одређује број табела које постоје у систему као 2M. Величина параметра N је најчешће два и користе се аутомати са слике 2.2.1.2.

* + 1. Bimodal предиктор

Група аутора је приметила да два различита скока могу имати исту глобалну историју што би код неких предиктора доводило до појаве негативне интерференције [7]. Негативна интерференција би се јављала уколико би ти скокови испољавали такво понашање да је једном од њих услов за скок често задовољен, а другом да није. Они су предложили систем за предвиђање скокова који скокове разврстава у две групе. Прва група су скокови код којих је услов скока често задовољен, а друга група су скокови код којих услов скока често није задовољен. Овај предиктор се састоји из три дела. Два дела представљају две табеле које чувају аутомате стања на основу којих се врши предвиђање, свака табела одговара једној групи скокова. Трећи део представља селектор који одабира која ће се од две табеле користити за предвиђање.

На слици 2.2.3.1. је приказана принципска шема овог система. Над најнижим битовима адресе скока и битовима из глобалног регистра историје примењује се операције ексклузивно или. Тако добијена вредност се користи за приступ до улаза у две табеле. Ти улази чувају аутомате стања на основу којих се даје предвиђање. Тако се добијају два предвиђања, а као коначно предвиђање узима се оно које одабере селектор. Један од улаза селектора се одабира на основу најнижих битова адресе скока. На основу вредности одабраног улаза се одређује које ће се предвиђање користити као коначно. Када се утврди коначан исход скока врши се ажурирање само оне табеле чије је предвиђање селектор одабрао као коначно. Селектор се ажурира након сваке инструкције скока.



Слика 2.2.3.1 Bimodal предиктор [7]

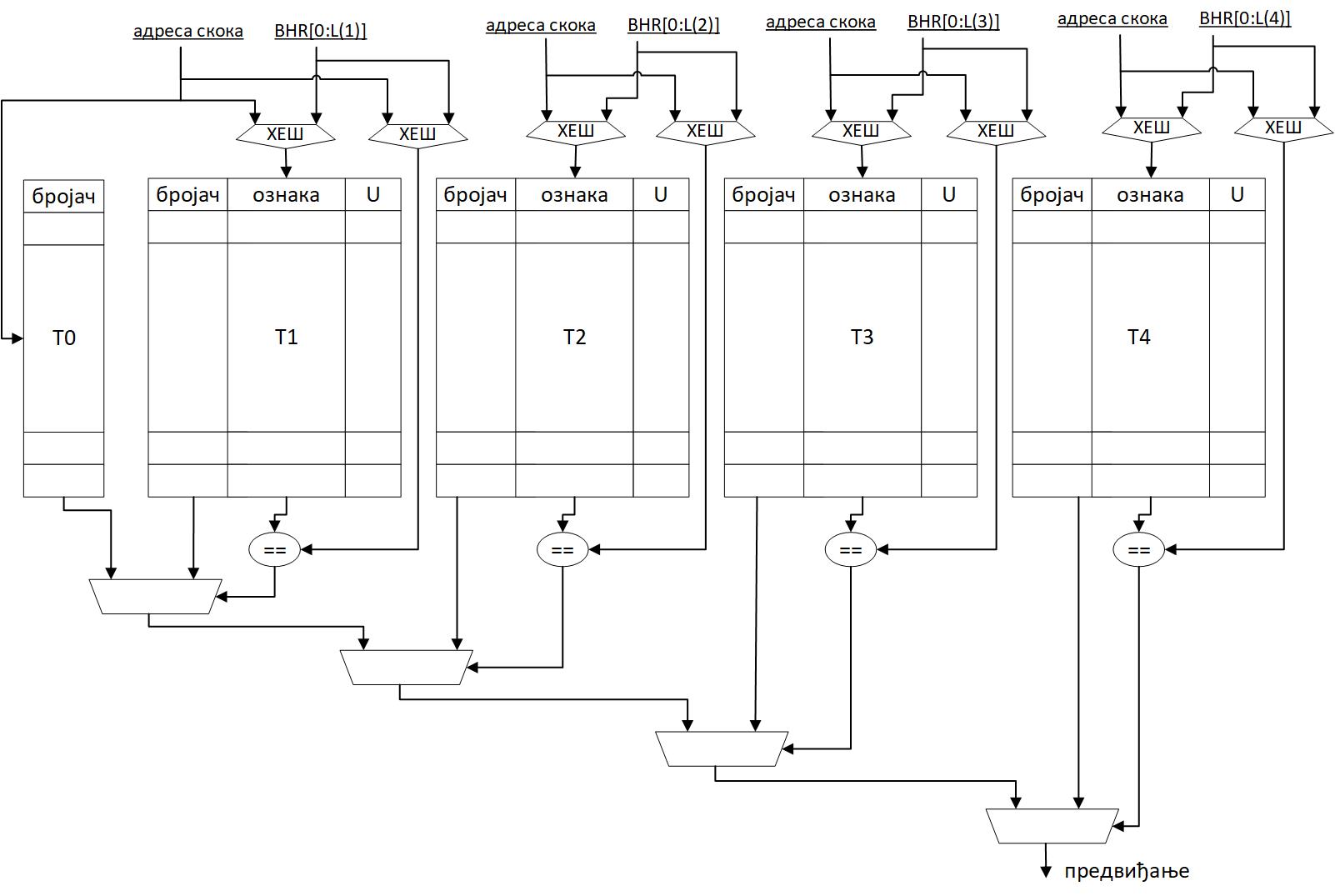
* + 1. TAGE предиктор

Група аутора [8] је предложила систем за предвиђање скокова који предвиђање рачуна на основу понашања скокова који су се скорије извршили, али и на основу скокова који су се давније извршили. Идеја TAGE (*TAgged GEometric history length*) предиктора је да користи различити број битова из глобалног регистра приликом рачунања предвиђања. Овај предиктор има неколико табела чији улази садрже бројаче на основу којих се одређује предвиђање. Поред бројача ови улази садрже и ознаке (*tags*) на основу којих се ове табеле претражују. Ознаке се добијају коришћењем хеш функције која као параметре користи бите из глобалног регистра историје и бите из адресе скока. За сваку од табела хеш функција користи различити број битова глобалног регистра историје. Приликом рачунања предвиђања табеле се претражују на основу израчунатих хеш функција. Уколико нека од табела садржи улаз са ознаком која има вредност једнаку израчунатој вредности хеш функције (погодак приликом претраге) онда се користи бројач из тог улаза и на основу њега одређује предвиђање. Уколико у више табела има погодака онда се предвиђање одређује на основу оне табела за чије је рачунање ознаке коришћено највише битова из глобалног регистра историје. Уколико ни у једној табели нема поготка тада се за предвиђање користи посебна табела (основни предиктор) којој се приступа на основу најнижих битова адресе скока. Улази ове табеле садрже бројаче на основу којих се одређује предвиђање.

На слици 2.2.4.1. је приказана принципска шема овог система. Са T0 је означен основни предиктор, а са *Т1-Т4* остале табеле које се претражују на основу ознаке коју рачуна хеш функција. За сваку табелу *Т1-Т4* приликом рачунања хеш функције користи се различити број битова из глобалног регистра историје (BHR) који су означени са *L(1)-L(4)*. Бројеви *L(1)-L(4)* формирају геометријски ред који су аутори описали једначином 2.2.4.1. Променљива α је константа која може имати различите вредности приликом формирања геометријског реда.

L(i) = (int)(αi-1 \* L(1) + 0.5) (2.2.4.1.)

Табела која даје коначно предвиђање назива се провајдер. Табела која има погодак и прва следећа је по броју битова глобалног регистра историје које користи назива се алтернативни провајдер. Уколико само једна табела има погодак, онда је алтернативни провајдер основни предиктор. Табеле имају још и поље *U* које представља колико је посматрани улаз користан за предикцију. Поље *U* провајдера се ажурира уколико се његово предвиђање разликује од предвиђања алтернативног провајдера. Уколико је тачно предвиђање провајдера поље *U* се инкрементира, а у супротном се декрементира. Поље бројач се ажурира само код оне табеле која је дала коначно предвиђање. Уколико је предвиђање тачно, бројач се инкрементира, а у супротном се декрементира.

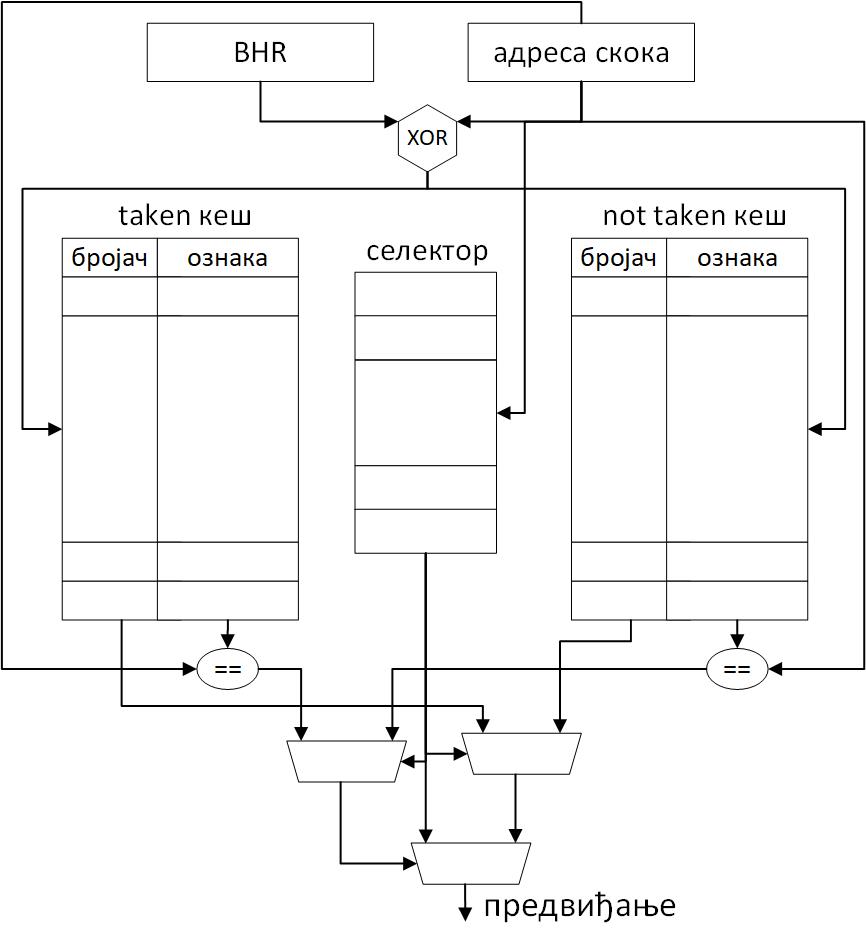


Слика 2.2.4.1. TAGE предиктор [8]

Уколико је предвиђање које је дато лоше тада се покушава пронаћи улаз у некој табели која користи већи број битова глобалног регистра историје и има вредност поља *U* једнаку нули. Ако такав улаз постоји у његово поље ознака се уписује резултат хеш функције, поље бројач се поставља на стање у зависности од коначног исхода скока, а поље *U* остаје на вредности нула, јер овај улаз још увек није коришћен. Уколико не постоји ни један улаз који има вредност поља *U* једнаку нули, тада се сва поља *U* свих табела декрементирају.

* + 1. YAGS предиктор

Систем за предвиђање скокова YAGS (*Yet Another Global Scheme*), који је описан у раду [1], представља комбинацију два предиктора *gshare* предиктора [9] и *bimodal* предиктора [7]. Комбинацијом поменутих предиктора добијен је овај хибридни предиктор који за циљ има да смањи интерференцију и редукује количину информација коју предиктор треба да чува током рада. Овај предиктор чине два *gshare* предиктора и један *bimodal* предиктор. Идеја је да два *gshare* предиктора служе за две групе скокова, слично као код *bimodal* предиктора. Један *gshare* предиктор се назива *taken* кеш, а други *not taken* кеш. Ова два предиктора понашају се попут кеш меморије и сваки њихов улаз има и поље ознака (*tag*). Улога *bimodal* предиктора је улога селектора, јер он треба да одабере коначно предвиђање. Такође, *bimodal* предиктор има улогу да даје предвиђање на основу тога да ли је често услов за скок био задовољен или не (да ли често долази до скока или не). Због тога *gshare* предиктори чувају само информације о оним скоковима чији се исход разликује од оног што је предвидео *bimodal* предиктор. Одабир коначног предвиђања започиње тако што се прво приступи *bimodal* предиктору и очита његово предвиђање. Ако је његово предвиђање да до скока долази тада се приступа *not taken* кешу како би се проверило да ли тај скок није имао скорије исход који се разликује од предвиђања које је очитано из *bimodal* предиктора. Ако постоји погодак унутар *not taken* кеша коначно предвиђање даје *not taken* кеш, а уколико нема поготка коначно превиђање је оно које је очитано из *bimodal* предиктора. Поступак је аналоган ако је очитано предвиђање *bimodal* предиктора да до скока не долази, само што се тада приступа *taken* кешу. На слици 2.2.5.1. је приказана принципска шема овог система.



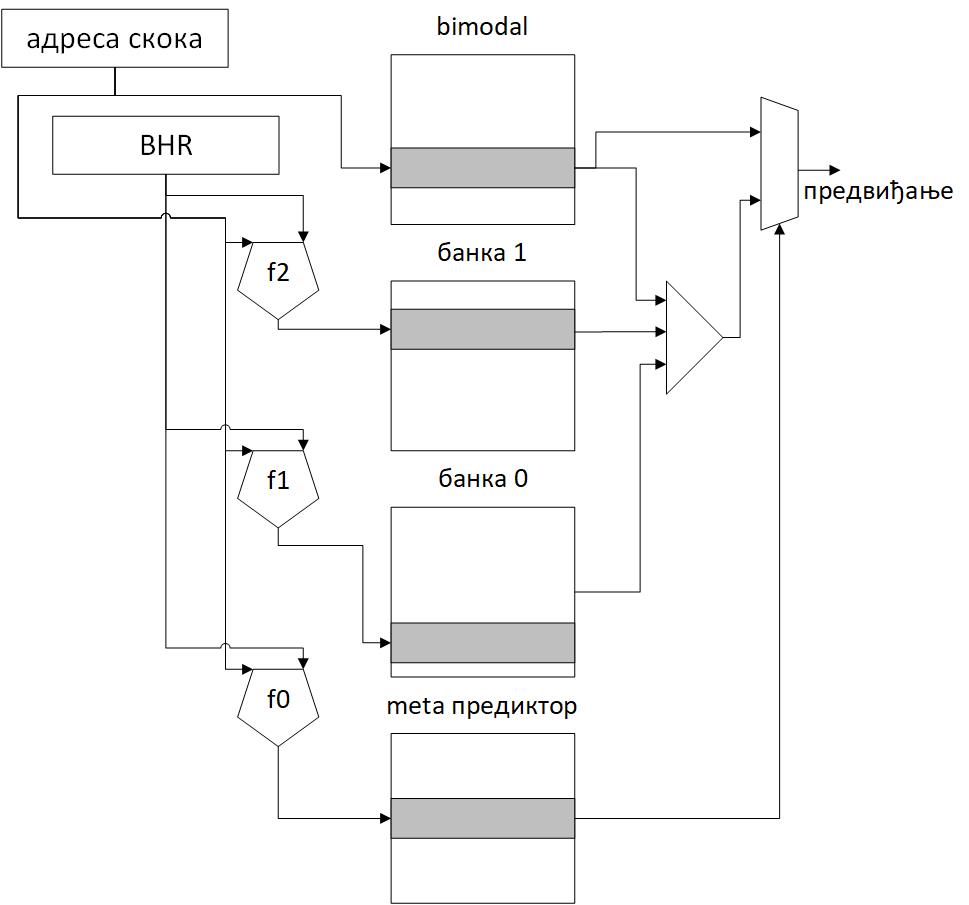
Слика 2.2.5.1. YAGS предиктор [1]

Приликом ажурирања овог система селектор се ажурира на исти начин као *bimodal* предиктор што је описано у поглављу 2.2.3. *Not taken* кеш се ажурира ако је као коначно предвиђање коришћено предвиђање које је он даo. Још се *not taken* кеш ажурира ако је селектор предвидео да до скока долази, а коначан исход скока је био супротан. *Taken* кеш се ажурира на сличан начин.

Како би ублажили проблем интерференције унутар *taken* и *not taken* кешева, аутори овог система су користили сет-асоцијативне кешеве. Сет-асоцијативни кешеви смањују појаву интерференције, али мора се користити додатни простор како би се водила евиденција о замени улаза. Код овог система користи се LRU (*Least Recently Used*) алгоритам замене.

* + 1. 2bc-gskew предиктор

Овај систем за предвиђање скокова представља хибридни предиктор који се састоји *e-gskew* предиктора [10] и *meta* предиктора, а описан је у раду [11]. Сам *e-gskew* предиктор се састоји од три банке (табеле). Једну банку представља *bimodal* предиктор, док друге две банке су табеле чији улази садрже аутомате стања на основу којих ове табеле дају предвиђање. *E-gskew* предиктор коначно предвиђање даје простим гласањем између банки (нпр. ако су две банке дале предвиђање да до скока долази, а једна да не долази, као коначно предвиђање се узима да ће до скока доћи). На овај начин се ублажава ефекат интерференције, јер уколико два скока деле исти улаз у некој банци, мања је вероватноћа да ће делити исти улаз у две или три банке. Да би се ово обезбедило за приступ банкама као индекси улаза се користе резултати различитих функција које као параметре користе адресу скока и вредност глобалног регистра историје. Ове функције су описане у радовима [10, 12]. На слици 2.2.6.1. је приказана принципска шема овог система.



Слика 2.2.6.1. 2bc-gskew предиктор [11]

На слици 2.2.6.1. се види да је улога *meta* предиктора да одабере коначно предвиђање између предвиђања које је дао *bimodal* предиктор и предвиђања које је дао *e-gskew* предиктор. Треба приметити да у систему постоји само један *bimodal* предиктор који је уједно и део *e-gskew* предиктора. Функције које рачунају индексе за приступ банкама су означене су означене са *f0*, *f1*, и *f2*.

Ажурирање овог система се обавља на такав начин да се ажурира само банка која је дала коначно предвиђање. Уколико је *meta* предиктор одабрао предвиђање које је дао *bimodal* предиктор, тада се ажурира само *bimodal* предиктор. Уколико је *meta* предиктор одабрао предвиђање које је дао *e-gskew* предиктор, тада се ажурирају све банке чије је предвиђање једнако коначном предвиђању. *Meta* предиктор се ажурира када су предвиђања *bimodal* предиктора и *e-gskew* предиктора различита.

* + 1. LVQ предиктор

Систем за предвиђање скокова LVQ (*Learning Vector Quantisation*) представља предиктор који користи неуралну мрежу LVQ за рачунање предикције [13]. Овај систем се састоји из два вектора. Један вектор одговара скоковима чији је услов за скок испуњен *Vt*, док други вектор одговара скоковима чији услов за скок није испуњен *Vnt*. Приликом рачунања предвиђања формира се и трећи вектор *X* чији су чланови најнижи битови из адресе скока и битови из глобалног регистра историје. Након формирања трећег вектора приступа се рачунању модификованих Хамингових дистанци (*Hamming distance*) између вектора *X* и два вектора *Vt* и *Vnt* по формулама 2.2.7.1. и 2.2.7.2.

HDt = ∑(Xi-Vti)2 (2.2.7.1.)

HDnt = ∑(Xi-Vnti)2 (2.2.7.2.)

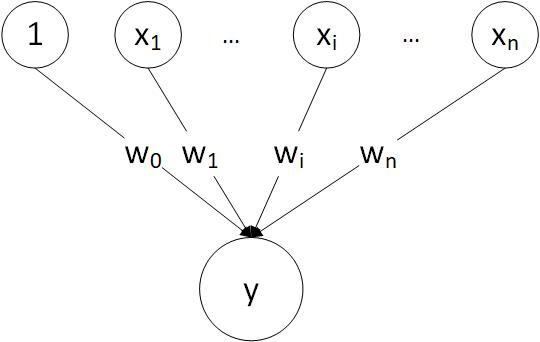
Хамингова дистанца која има мању вредност одређује предвиђање. Уколико мању вредност има *HDt* тада је предвиђање да је услов за скок испуњен. Уколико мању вредност има *HDnt* тада је предвиђање да услов за скок није испуњен.

Ажурирање овог система се обавља тако што се ажурира само онај *V* вектор чија је Хамингова дистанца била мања, тј. онај вектор на основу кога је одређено предвиђање. Ажурирање вектора се обавља по формули 2.2.7.3. где је *а(t)* константа која представља фактор учења и има вредност мању од 0,1. Уместо знака *#* примењује се или операција сабирања или операција одузимања у зависности од тога да ли је предвиђање исправно или погрешно, респективно.

V(t+1)=V(t) # a(t)[X(t)-V(t)] (2.2.7.3.)

* + 1. Perceptrons предиктор

Овај систем за предвиђање скокова је заснован на коришћењу неуралне мреже са перцептронима (*perceptrons)*  и описан је у раду [14]*.* Аутори овог система су се одлучили за коришћење ове неуралне мреже због њене лаке имплементације у хардверу. У основи ове неуралне мреже налази се перцептрон који је приказан на слици 2.2.8.1. Перцептрон се може представити вектором чији су елементи тежине (*weights*) *w0..n* које служе да на основу улазног вектора *x* одреде излазну вредност перцептрона *y*.



Слика 2.2.8.1. Изглед перцептрона

Коначна вредност излаза *y* се рачуна по формули 2.2.8.1. Вектор *x* као своје елементе садржи вредности на основу бита из глобалног регистра историје. Ако бит глобалног регистра историје има вредност нула, онда одговарајући елемент вектора *x* има вредност минус један, у супротном има вредност један. Приметити да је елемент *x0* увек једнак јединици тако да тежина *w0* представља утицај на предвиђање које се односи на претходна извршавања тог скока.

y=w0+∑xi\*wi (2.2.8.1.)

Коначно предвиђање се одређује на основу срачунате вредности *y*. Уколико је вредност *y* позитивна предвиђање је да до скока долази. Уколико је вредност *y* негативна предвиђање је да до скока не долази.

Ажурирање перцептрона се обавља када буде познат коначан исход скока ако је један од два услова задовољен. Први услов је да се предвиђање разликује од коначног исхода скока, а други услов је да је апсолутна вредност *y* мања од границе која означава да ли је неурална мрежа довољно научила. Ажурирају се сви елементи вектора *w* према формули 2.2.8.2. Са *t* је означен исход скока и има вредност један или минус један у зависности да ли је услов скока био задовољен или не.

wi=wi+t\*xi, i=0..n (2.2.8.2)

Овај систем се састоји од вектора перцептрона. Када се појави инструкција скока тада се на основу битова из адресе скока приступа вектору са перцептронима. На овај начин одабрани перцептрон се користи за одређивање предвиђања.

* + 1. Помоћни предиктори

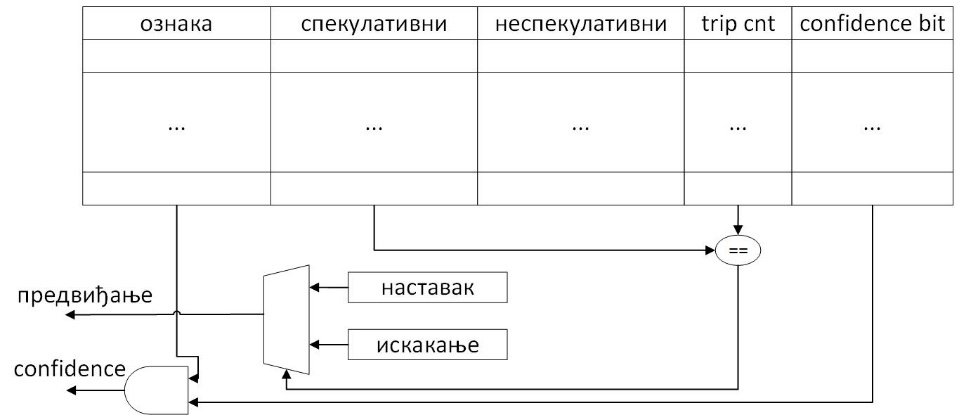
Помоћни предиктори (*side predictors*) су мали специјализовани предиктори за скокове чије је исходе тешко предвидети (*hard*-*to-predict branches*) [2]. Ови предиктори се користе у комбинацији са неким већим предикторима (главни предиктори). Њихова улога је да одреде предвиђања за скокове чије исходе већи предиктори лоше одређују.

* + - 1. LTP предиктор

LTP (*Loop Termination Predictor*) је систем за предвиђање завршетка програмских петљи и описан је у раду [15]. Аутори овог система су желели да повећају прецизност главног предиктора тако што би LTP предиктор давао предвиђање о томе када се завршава петља (тренутак када се искаче из петље). Код главних предиктора проценат промашаја искакања из петље је обрнуто пропорционалан броју итерације петље, а како се у програмима петље често појављују, овај помоћни предиктор може да поправи прецизност. Идеја овог предиктора је да препозна да се ради о инструкцији искакања из петље и да предвиди тренутак кад ће се искочити из петље. Аутори су споменули два начина на основу којих се може утврдити да ли се ради о таквим скоковима. Први начин је да преводилац такве инструкције скока посебно означи, а други начин је да се посматра одредишна адреса скока и ако је то скок уназад може се сматрати да се ради о петљи у програму.

Предиктор се састоји из једне табеле чији улази имају пет поља и приказан је на слици 2.2.9.1. Два поља су бројачи који броје колико се пута скок десио (колико је пута услов био задовољен), један се назива спекулативни (*speculative iter*), а други неспекулативни (*non-speculative iter*). Затим поље које чува број који представља колико је пута услов за скок био задовољен од последњег пута када није био задовољен (*trip counter*). Једно поље које чува информацију о томе да ли се два пута за редом исти број пута (*trip counter* садржи тај број) поновио скок (*confidence bit*). Пошто се у табели чувају информације о више различитих скокова, постоји поље које представља ознаку (*tag*) тих инструкција скока на основу којих се табела претражује. Када наиђе инструкција скока тада се приступа овој табели. Уколико има поготка врши се поређење поља спекулативног бројача са пољем *trip counter.* Ако су једнаки проверава се да ли је поље *confidence bit* постављено на јединицу. Ако јесте онда LTP даје предвиђање да се из петље искаче. Ако поља спекулативног бројача и *trip counter* нису једнака тада се вредност спекулативног бројача инкрементира.

У тренутку кад се зна коначан исход скока, ако се ради о инструкцији скока којом се искаче из петље, ако је главни предиктор промашио предвиђање и ако те инструкције нема у табели тада се она убацује у табелу и сва поља се постављају на вредност нула. Ако је инструкција скока у табели и услов скока је испуњен тада се неспекулативни бројач за тај скок инкрементира. Ако је инструкција скока у табели и услов скока није испуњен тада се провера једнакост неспекулативног бројача и *trip counter*. Уколико су једнаки поље *confidence bit* се поставља на јединицу, а у супротном се поставља на нулу. Након тога се неспекулативни бројач инкрементира и његова вредност се копира у поље *trip counter* и потом се неспекулативни бројач поставља на нулу.



Слика 2.2.9.1. LTP предиктор [15]

* + - 1. BMP предиктор

Систем за предвиђање скокова BMP (*Branch Misprediction Predictor)* представља помоћни предиктор и описан је у раду [16]. Идеја аутора је да овај предиктор врши предвиђање оних скокова који умањују перформансе процесора, тј. оних скокова за које главни предиктор често даје погрешна предвиђања. Овај предиктор прати историју погрешних предвиђања и на основу ње предвиђа који ће бити следећи скок чије ће предвиђање главни предиктор да одреди погрешно. Kада препозна да се ради о поновном доласку на извршавање инструкције скока чије је предвиђање главни предиктор претходно одредио погрешно, тада BMP врши корекцију предвиђања. Користи бројаче који воде евиденцију о томе колико се инструкција скока извршило између два погрешна предвиђања главног предиктора.

LTP предиктор након периода учења и уколико је број итерације петљи константан у време извршавања добро побољшање прецизности главног предиктора. Међутим, он је осетљив на промену броја итерација у време извршавања. За разлику од њега BMP предиктор показује добре резултате уколико број итерација није константан или уколико се превремено прекида петља коришћењем наредбе *break* [2]*.*

Литература

1. Eden, Avinoam N., and Trevor Mudge. "The YAGS branch prediction scheme." *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society Press, 1998.
2. Mittal, Sparsh. "A Survey of Techniques for Dynamic Branch Prediction." *arXiv preprint arXiv:1804.00261* (2018).
3. Loh, Gabriel H. "Revisiting the performance impact of branch predictor latencies." *Performance Analysis of Systems and Software, 2006 IEEE International Symposium on*. IEEE, 2006.
4. Yeh, Tse-Yu, and Yale N. Patt. "Two-level adaptive training branch prediction." *Proceedings of the 24th annual international symposium on Microarchitecture*. ACM, 1991.
5. Yeh, Tse-Yu, and Yale N. Patt. "A comparison of dynamic branch predictors that use two levels of branch history." *ACM SIGARCH Computer Architecture News*. Vol. 21. No. 2. ACM, 1993.
6. Pan, Shien-Tai, Kimming So, and Joseph T. Rahmeh. "Improving the accuracy of dynamic branch prediction using branch correlation." *ACM Sigplan Notices*. Vol. 27. No. 9. ACM, 1992.
7. Lee, Chih-Chieh, I-Cheng K. Chen, and Trevor N. Mudge. "The bi-mode branch predictor." *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society, 1997.
8. Seznec, André and Pierre Michaud. “A case for (partially) TAgged GEometric history length branch prediction.” *J. Instruction-Level Parallelism* 8 (2006): n. pag.
9. McFarling, Scott. *Combining branch predictors*. Vol. 49. Technical Report TN-36, Digital Western Research Laboratory, 1993.
10. Michaud, Pierre, André Seznec, and Richard Uhlig. "Trading conflict and capacity aliasing in conditional branch predictors." *ACM SIGARCH Computer Architecture News* 25.2 (1997): 292-303.
11. Seznec, André, and Pierre Michaud. *De-aliased hybrid branch predictors*. Diss. INRIA, 1999.
12. Seznec, André, and Francois Bodin. "Skewed-associative caches." *International Conference on Parallel Architectures and Languages Europe*. Springer, Berlin, Heidelberg, 1993.
13. Egan, Colin, et al. "Two-level branch prediction using neural networks." *Journal of Systems Architecture* 49.12-15 (2003): 557-570.
14. Jiménez, Daniel A., and Calvin Lin. "Dynamic branch prediction with perceptrons." *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on*. IEEE, 2001.
15. Sherwood, Timothy, and Brad Calder. "Loop termination prediction." *International Symposium on High Performance Computing*. Springer, Berlin, Heidelberg, 2000.
16. Sendag, Resit, et al. "Low power/area branch prediction using complementary branch predictors." *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008.
17. Computer Architecture, Computer and Informatics Science University of Pennsylvania (<https://www.cis.upenn.edu/~milom/cis501-Fall12/traces/trace-format.html>, 15.08.2018.)
18. Championship Branch Prediction CBP-5 (<https://www.jilp.org/cbp2016/>, 15.08.2018.)
19. Jiménez, Daniel A., Stephen W. Keckler, and Calvin Lin. "The impact of delay on the design of branch predictors." *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*. ACM, 2000.
20. Jin, Wenbing, et al. "A novel architecture for ahead branch prediction." *Frontiers of Computer Science* 7.6 (2013): 914-923.