ՏԵՂԵԿԱՏՎԱԿԱՆ ՏԵԽՆՈԼՈԳԻԱՆԵՐ, ԷԼԵԿՏՐՈՆԻԿԱ, ՌԱԴԻՈՏԵԽՆԻԿԱ

ԻՆՖՈՐՄԱՏԻԿԱ ԵՎ ՀԱՇՎՈՂԱԿԱՆ ՏԵԽՆԻԿԱ

ՀՏԴ 681.3

Ա.Կ. ԹՈՒՄԱՆՅԱՆ, Ա.Գ. ՔԱՄԱԼՅԱՆ

ՊԱՅՄԱՆԱԿԱՆ ԱՆՑՈՒՄՆԵՐԻ ԵՎ ՊԱՅՄԱՆԱԿԱՆ ՀՐԱՄԱՆՆԵՐԻ ԿԱԶՄԱԿԵՐՊՄԱՆ ՄԵԹՈԴՆԵՐԻ ՀԵՏԱԶՈՏՈՒՄԸ ՄԻԿՐՈՊՐՈՑԵՍՈՐԱՅԻՆ ՀԱՄԱԿԱՐԳԵՐՈՒՄ

Դիտարկվում է ծրագրերում ճյուղավորման իրականացումը կազմակերպելու երկու մեթոդ՝ անցման հրամանների և պայմանական (պրեդիկատային) հրամանների կիրառմամբ: Մշակվել է պրոցեսորի հրամանների համակարգը, որը ներառում է պայմանական հրամաններ: Մշակվել է պայմանական հրամանների կատարման կոնվեյերի սխեման՝ հրամանի կատարման (EX) բլոկի համար: Կատարվել է կոնվեյերի տակտերի քանակի գնահատում՝ անցման հրամանների և պայմանական հրամանների կիրառմամբ: Դիտարկվել է Loop unrolling (ցիկլերի փոխակերպում) տեխնոլոգիան:

Առանցքային բառեր. կոնվեյեր, կանխատեսման սխեմա, պրեդիկտոր, պրեդիկատային հրամաններ, Loop unrolling:

Ներածություն: Իրերի ինտերնետի շնորհիվ մեծացել է ինչպես պրոցեսորների քանակը, այնպես էլ անհրաժեշտ փոխհամաձայնեցումը չիպի չափի, հզորության, արժեքի և արտադրողականության միջև: Միկրոկոնտրոլերների նախագծման դեպքում օգտագործվում են CISC և RISC ճարտարապետությունները, RISC ճարտարապետությունն ավելի տարածված է:

Առավելագույն արտադրողականության և հրամանների զուգահեռ կատարման հնարավորություններին հասնելու համար միկրոկոնտրոլերներն օգտագործում են Հարվարդյան ճարտարապետություն՝ առանձնացված հիշողությամբ և առանձնացված ծրագրի և տվյալների շինաներով:

Պրոցեսորի աշխատանքի արտադրողականության բարձրացման հիմնական միջոցը հաշվարկների կոնվեյերացումն է: Ավելի արդյունավետ է հրամանային ցիկլի կոնվեյերացման կիրառումը:

Իդեալական կոնվեյերային պրոցեսորը պետք է ունենա CPI (Cycles per instruction), որը հավասար է՝ CPI=իդեալական CPI + structural stalls + data hazards stalls + control hazards stalls: Կոնվեյերի արտադրողականության վրա ամենամեծ ազդեցություն ունեն ղեկավարման կոնֆլիկտները:

Անցումների ստատիկ և դինամիկ կանխատեսում: Անցումների կանխատեսումը ժամանակակից պրոցեսորներում կառավարման կոնֆլիկտների դեմ պայքարի ամենաարդյունավետ միջոցն է: Անցումների կանխատեսումն օգտագործվում է կոնվեյերը լցնելու համար: Ընդ որում՝ կիրառվում է անցումների կանխատեսման երկու մեթոդ՝ ստատիկ և դինամիկ:

Պարզ ստատիկ կանխատեսման մեթոդներ.

- միշտ անցում կա (always taken) – կանխատեսման ճշգրտությունը 60-70%,
- միշտ անցում չկա(always not taken) - կանխատեսման ճշգրտությունը 30-40%
- անցում հետ՝ կա (backwards), անցում առաջ՝ չկա:

Կանխատեսման ճշգրտությունը ճիշտ կանխատեսված անցման հրամանների հարաբերությունն է կանխատեսումների ընդհանուր թվին:

Ժամանակակից ընդհանուր նշանակությամբ պրոցեսորներն օգտագործում են կանխատեսումների հետևյալ մեթոդները.

Կոմպիլյատորի տեխնոլոգիաներ. պրոֆիլավորում՝ հիմնված ծրագրի վրա, և հուշումներ ծրագրավորողի կողմից:

Կոմպիլյատորի տեխնոլոգիաների թերություն. այս մեթոդները չեն կարող հարմարվել անցման հրամանների կատարման դինամիկային:

Ստատիկ կանխատեսումը կատարվում է հրամանների վերծանման փուլում: Անցումների դինամիկ կանխատեսումը հիմնված է յուրաքանչյուր անցման հրամանի անցման կատարման մասին պատմության կուտակման վրա: Անցումների դինամիկ կանխատեսումն ապահովում է անցումների կանխատեսման ավելի մեծ ճշգրտություն:

Կանխատեսվող ճյուղի կոդը կատարվում է չարաշահմամբ, այսինքն՝ կատարման արդյունքները չեն ամրագրվում ընդհանուր նշանակությամբ ռեգիստրներում և հիշողության մեջ: Չարաշահմամբ կանխատեսումը և՛ ILP-ի կատարողականի, և՛ անարդյունավետության աղբյուր է:

Երբ ճյուղավորման կանխատեսումը կատարյալ է, չարաշահությունը բարձրացնում է արտադրողականությունը և մի փոքր մեծացնում էներգիայի սպառումը, նույնիսկ կարող է խնայել էներգիան, բայց երբ ճյուղերը կանխատեսվում են սխալ, պրոցեսորը պետք է բեռնաթափի հաշվարկները, և այդ դեպքում ամբողջ աշխատանքն ու էներգիան վատնվում են ապարդյուն: Ճյուղավորման սխալ կանխատեսման տույժը CPI-ի բարձրացման հիմնական պատճառն է [1]:

Ժամանակակից ներկառուցված պրոցեսորները մեծապես կախված են ճյուղավորումների կանխատեսումների ճշգրտությունից: Խոշոր կանխատեսիչները՝ բարդ կանխատեսման ալգորիթմներով, առաջացնում են ավելի շատ էներգիայի սպառում, ինչը անցանկալի է ներկառուցված համակարգերի դեպքում: Ներկա-

ռուցված համակարգերում օգտագործվող միկրոկոնտրոլերներն ունեն կարճ կոնվեյեր: Հիմնականում դա պայմանավորված է դրանց վրա կատարվող ծրագրերով և էներգիայի սպառման կրճատմամբ (որքան փոքր է կոնվեյերի երկարությունը, այնքան ցածր է սխալ կանխատեսման համար նախատեսված տույժը): Ճարտարապետական մակարդակում էլեկտրաէներգիայի սպառումը նվազեցնելու համար կարելի է օգտագործել 3 մոտեցում.

• Առաջին մոտեցումը հիմնված է անցումային հրամանների քանակի կրճատման վրա:

• Երկրորդը հիմնված է ավելի պարզ կանխատեսման սխեմաների կիրառման վրա:

• Երրորդ մոտեցումը ստատիկ կանխատեսմամբ ավելի շատ քանակով անցումներ կանխատեսելն է: Սա նվազեցնում է դինամիկորեն կանխատեսվող հրամանների քանակը, ինչը հանգեցնում է կանխատեսման ճշգրտության նվազմանը:

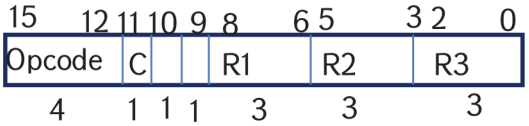
Անցումային հրամանները, որոնք ունեն դինամիկ կանխատեսող ռեսուրսներից առավել ճշգրիտ կանխատեսում, կանխատեսվում են դինամիկորեն: Այս մոտեցումը թույլ է տալիս պարզեցնել դինամիկ կանխատեսման սխեման, քանի որ այդ դեպքում պահանջվում են ավելի փոքր չափերով BTB և PHT [1]:

Անցման հրամանների կրճատման մեթոդները: Անցման հրամանների քանակի կրճատման համար կարելի է կիրառել երկու մեթոդ՝ պայմանական հրամաններ և ցիկլերի փոխակերպում: Դիտարկենք, թե ինչպես են այս երկու մեթոդներն ազդում կոնվեյերի աշխատանքի վրա: Պահանջվում է մշակել ոչ մեծ միկրոպրոցեսոր՝ կոնվեյերային ճարտարապետությամբ [2]:

1. Պրոցեսորի հրամանների համակարգի մշակումը, որը ներառում է պայմանական հրամաններ: Պրոցեսորի մշակումը սկսվում է պրոցեսորի հրամանների հավաքածուի մշակմամբ: Այս փուլում որոշվում են հրամանների հավաքածուն, տվյալների տեսակներն ու չափերը, հրամանների ձևաչափերը, պրոցեսորի ներքին հիշողության մոդելները և օպերանդների հասցեավորման մեթոդները: Պրոցեսորը ներառում է ընդհանուր նշանակության 8 ռեգիստր: GPRs-ը (General Purpose Registers) ներկայացնում է եռափորթ հիշողություն՝ երկու փորթը կարդալու և մեկ փորթը գրելու համար:

Պրոցեսորի թվաբանական-տրամաբանական սարքը (ԹՏՍՍ) կատարում է 16-կարգանի նշանով ամբողջ թվերով թվաբանական գործողություններ և 16-կարգանի օպերանդներով տրամաբանական գործողություններ: Օպրատիվ հիշողությունը (SRAM) ունի 256x16 ծավալ: Օպրատիվ հիշողությունում գտնվում են կատարողական ծրագիրը և տվյալները: Գրանցման և ընթերցման համար շինաներն առանձնացված են: Գրանցումը սինքրոն գործողություն է, իսկ ընթերցումը՝ ասինքրոն:

Պրոցեսորի ռեգիստրները՝
Reg PC- ծրագրային հաշվիչ (Program Counter).
Reg IR – հրամանի ռեգիստր (Instruction Register).
Reg Flags – պրոցեսորի վիճակի ռեգիստր:
Հրամանների ձևաչափը ներկայացված է նկ.1-ում:



Նկ. 1. Հրամանի ձևաչափը

Հրամանների համակարգը ներկայացված է աղ. 1-ում: Բիթ C-ն որոշում է կատարվող հրամանը, հանդիսանում է պրեդիկատային (եթե C=1) կամ դա սովորական հրաման (եթե C=0): Պայմանի կոդը երկբիթանի է: Վիճակի ռեգիստրն ունի 3 դրոշակ՝ ZF,SF,CF (աղ.1).

Աղյուսակ 1

Oper code	**C**	**Cond. code**		
0 0 0 0	**0**	-	**Nop**	**Գործողություն չկա**
0 0 0 1	**0**	-	**Load r1, mem**	**Օպերանդի բեռնավորում ռեգիստր**
0 0 1 0	**0**	-	**store mem, r1**	**Օպերանդի գրանցում հիշողություն**
0 0 1 1	**0**	-	**add r1,r2,r3**	**Գումարում**
	1	**00**	**add.equ**	**Գումարում, եթե ZF=1**
	1	**10**	**add.g**	**Գումարում, եթե (SF==0)&&(ZF!=0)**
	1	**11**	**add.g_eq**	**Գումարում, եթե արդյունքը >=0**
0 1 0 0			**sub r1,r2,r3**	**Հանում**
. . . .				
1 1 1 1			**I/O**	

add,sub, mul հրամանները կարող են կատարվել պայմանականորեն (C=1) և անպայմանորեն (C=0): Պայմանի կոդի հաջորդ երկու բիթերը (cond. code) ցույց են տալիս սահմանվող պայմանը:

Պայմանական հրամանների առկայությունը թույլ է տալիս նվազեցնել պայմանական անցումների քանակը, և հետևաբար՝ կնվազի նաև սխալ կանխատեսված հրամանների (mispredictions) քանակը: Սխալ կանխատեսումների (mispredictions) կրճատումը հանգեցնում է էներգիայի սպառման նվազմանը, քանի որ կրճատվում է կոնվեյերի վերաբեռնումների քանակը: Պայմանական հրամանների առկայությունը, պայմանի չկատարելու դեպքում, չի հանգեցնում կոնվեյերի վերաբեռնմանը [3]:

2. Անցումների կանխատեսման սխեմայի մշակումը: Անցումային կանխատեսման սխեմաները պարզեցնելու համար կօգտագործենք Սմիթի կանխատեսման պարզ կանխատեսիչներ:

BTB-ի կազմակերպման մեթոդների վերլուծություն: Դինամիկ կանխատեսումը տեղի է ունենում հրամանների ընտրման փուլում: BTB-ի կառուցվածքը նման է լրիվ ասոցիատիվ քեշ-հիշողության (Fully Associative Cache-ի) կառուցվածքին: BTB-ն փաստացիորեն հանդիսանում է քեշ, որը պարունակում է տեղեկություն անցման հրամանների մասին: BTB-ն բաղկացած է երկու մասից՝ տեգերի (tag) և SRAM հիշողություններից: Տեգերը հանդիսանում են անցման հրամանի հասցեի մի մասը: Յուրաքանչյուր կատարված անցման հրամանին համապատասխանում է BTB-ի մեկ տող (Tag): PC-ում տեղակայված հրամանի տեգը համեմատվում է BTB-ում անցման հրամանների տեգերի հետ: Եթե դրանք համընկնում են, ապա վերլուծվում են կանխատեսման բիթերը (prediction bits, PHT) և փոխանցվում պրեդիկտորին: Եթե պրեդիկտորը կանխատեսում է, որ անցում կլինի, ապա Icache-ին տրվում է կանխատեսված անցման հասցեն (Brahch Target Address): Եթե պրեդիկտորը կանխատեսում է, որ անցում չի լինի, ապա ICache -ին տրվում է PC+4 հասցեն: Այսպիսով, BTB-ն կատարում է հետևյալ ֆունկցիաները.

1. Կանխատեսում է՝ արդյոք հաջորդը անցման հրաման է:
2. Կանխատեսում է՝ անցում կլինի, թե ոչ:
3. Կանխատեսում է անցման հասցեն:

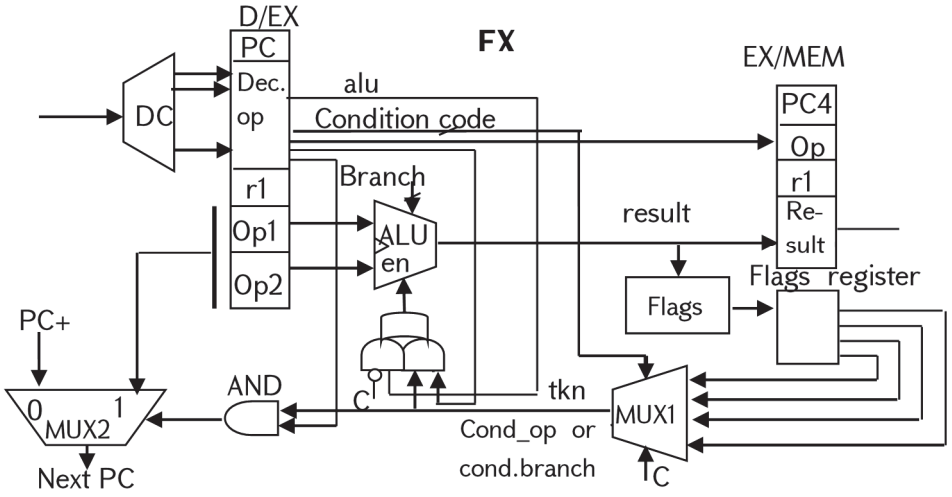
BTB-ում պատմություն չունեցող անցման հրամանները կանխատեսվում են ստատիկ կանխատեսման ալգորիթմի միջոցով: BTB-ի դիտարկված տարբերակում PHT-ն ներառված է BTB-ի կազմում: Այս դեպքում ենթադրվում է, որ BTB-ում գտնվում են անցման հրամանները, նույնիսկ եթե անցում ոչ մի անգամ չեն կատարել: Այս դեպքում BTB-ի ծավալը կարող է բավականին մեծ լինել: Մյուս մոտեցումն այն է, որ BTB-ում տեղադրվում են միայն այն հրամանները, որոնք ավարտվել են անցումով: Այս դեպքում PHT-ն պետք է պահվի առանձին [4]:

Ընդհանուր նշանակությամբ պրոցեսորներն օգտագործում են երկար կոնվեյերներ: Դրանք Intel, AMD պրոցեսորներ են (ճարտարապետությունx86-64), ARM պրոցեսորների ավելի հին մոդելներ: Որոշ պրոցեսորների, ինչպիսին է Alder Lake-ը, կոնվեյերի երկարությունը կազմում է 19 փուլ:

Միկրոկոնտրոլերներում օգտագործվում են կարճ կոնվեյերներ (մինչև 5 փուլ): Նման պրոցեսորներում նպատակահարմար չէ օգտագործել մեծ ծավալով BTB: Ցանկալի է օգտագործել անցումների կանխատեսման այլ մոտեցում: BTB-ում պահել տեղեկություն միայն այն անցման հրամանների մասին, որոնք ավարտվել

են հաջողությամբ (այսինքն՝ taken): Ավելին, եթե BTB-ից որևէ հրաման չի ավարտվում անցումով (not taken), ապա այն հեռացվում է BTB-ից՝ դրանով իսկ ազատելով տարածք այլ հրամանների համար: Կանխատեսման ճշտության ստուգումը տեղի է ունենում EX փուլում՝ նախորդ հրամանի կատարումից հետո: Սխալ կանխատեսման դեպքում կոնվեյերը վերաթողարկվում է սխալ կանխատեսման հրամանից սկսած՝ (misprediction) այլընտրանքային ճյուղի վրա: Կոնվեյերի փուլերը մաքրվում են սխալ ճյուղի հրամաններից: Պրոցեսորի ներքին վիճակը նույնպես պետք է վերականգնել այն վիճակին, որն ուներ մինչև սխալ հասկացված ճյուղը՝ լրացուցիչ ժամանակի և էներգիայի գնով: Քանի որ կոնվեյերը 5-աստիճանի է, ուստի վիճակի վերականգնման համար կպահանջվի *3 տակտ*: Անցման հրամանները կազմում են կատարվող հրամանների մոտավորապես 25%-ը: Տեսականորեն կոնվեյերի վրա կարող են լինել միաժամանակ 2 անցման հրամաններ [5]:

Պայմանական հրամանների կատարման համար 5-աստիճան կոնվեյերի վրա հրամանի կատարման (EX) բլոկի մշակումը: Նկ. 2-ում ներկայացված է թվաբանական և տրամաբանական պայմանական հրամանների և պայմանական ու անպայման անցումների հրամանների կատարման (execution) բլոկի կառուցվածքը:



Նկ. 2. Հրամանի կատարման բլոկ

Այս բլոկի մուտքին հրամանը տրվում է DC փուլից: EX փուլում կատարվում են անպայման և պայմանական թվաբանական և տրամաբանական, ինչպես նաև անցման հրամանները: Անցման հրամանները պրեդիկատային չեն: Պրեդիկատային հրամանները կատարվում են ԹSU-ում միայն այն դեպքում, եթե պայմանը կատարվում է: Ոչ պրեդիկատային հրամանները կատարվում են միշտ:

En ազդանշանը թույլատրում է ԹSU-ի աշխատանքը:

En = C&Taken +~C&alu:

Կոնվեյերի աշխատանքը պայմանական հրամաններով: Որպես օրինակ դիտարկենք ամենամեծ ընդհանուր բաժանարարի որոշումը (Էվկլիդեսի ալգորիթմը)՝ հիմնված հանման վրա:

C լեզվով

```
while (i!= j)
   {
     if (i > j)
        i -= j;
     else
        j -= i;
   }
```

Սովորական ասեմբլերով

```
gcd     cmp  r0, r1 ;      reached the end?
        beq      stop
        blt      less       ;  if r0 < r1
        sub   r0, r0, r1    ;  r0-r1
        bal       gcd
less    sub    r1, r1, r0    ; else  r1-r
        bal  gcd    ;    uncond. branch
stop
```

Պայմանական հրամաններով ասեմբլեր՝

```
gcd cmp r0, r1 ;
subgt r0, r0, r1 ; r0-r1
sublt r1, r1, r0 ; else r1-r0
bne gcd ; reached the end?
stop
```

Կոդից երևում է, որ պայմանական հրամանների կամ պրեդիկացիայի օգտագործումը թույլ է տվել ամբողջությամբ խուսափել ճյուղավորումից else և then օպերատորներում: Եթե r0-ը և r1-ը հավասար են, ապա SUB հրամաններից ոչ մեկը չի կատարվի: Յուրաքանչյուր ցիկլի սկզբում while-ի ստուգում իրականացնող անցման հրամանի անհրաժեշտությունը վերանում է:

Գնահատենք կոնվեյերի տակտերի քանակը, ինչը պահանջվում է վերը նշված ծրագրերն իրականացնելու համար.

1. Կոնվեյերի տակտերի քանակի գնահատում սովորական ասեմբլերով ծրագրի դեպքում: Օրինակ՝ A=56, B=32: r0=56, r1=32:

```
1). gcd cmp r0, r1 ; end?
2). beq stop No
3). blt less ; if r0 < r1
4). sub r0, r0, r1 ; r0=24, r1=32
5). bal gcd
6). gcd cmp r0, r1 ; end?
7). beq stop No
8). blt less ; if r0 < r1
9). less sub r1,r1, r0 ; r1=8, r0=24
10. bal gcd
11. gcd cmp r0, r1 ; end?
12). beq stop No
```

13) blt less ; if r0 < r
14). sub r0, r0, r1 ; r0=24-8=16; r1=8
15). bal gcd
16). gcd cmp r0, r1 ; end?
17). beq stop No
18). blt less ; if r0 < r1
19). sub r0,r0, r1 ; r0=16-8=8, r1=8
20). bal gcd
21). gcd cmp r0, r1 ; end?
22). beq stop Yes
23) stop

Յուրաքանչյուր հրամանի կատարման ընթացքում կոնվեյերի վիճակը ներկայացված է աղ. 2-ում:

Աղյուսակ 2

	IF	ID	EX	MEM	WB
1.	cmp	-	-	-	-
2.	beq	cmp	-	-	-
3.	blt	beq	cmp	-	-
4.	sub	blt	beq	cmp	-
5.	bal	sub	blt	beq	Cmp
6.	cmp	bal	sub	blt	Beq
7.	beq	cmp	bal	sub	Blt
8.	blt	beq	cmp	bal	Sub
9.	sub	blt	beq	cmp	Bal
10.	bal	sub	blt	beq	Cmp
11.	cmp	bal	sub	blt	Beq
12.	beq	cmp	bal	sub	Blt
13.	blt	beq	cmp	bal	Sub
14.	sub	blt	beq	cmp	Bal
15.	bal	sub	blt	beq	Cmp
16.	cmp	bal	sub	blt	Beq
17.	beq	cmp	bal	sub	Blt
18.	blt	beq	cmp	bal	Sub
19.	sub	blt	beq	cmp	Bal
20.	bal	sub	blt	beq	Cmp
21.	cmp	bal	sub	blt	beq
22.	beq	cmp	bal	sub	Blt
23.	blt	beq	cmp	bal	Sub
24.	sub	blt	beq	cmp	Bal
25.	bal	sub	blt	beq	Cmp

Աղյուսակ 2-ի շարունակությունը

	IF	ID	EX	MEM	WB
26.	cmp	bal	sub	blt	Beq
27.	beq	cmp	bal	sub	Blt
28.	stop	beq	cmp	bal	Sub
29.		stop	beq	cmp	Bal
30.			stop	beq	Cmp
31.				stop	Beq
32.					Stop

Սկսած ծրագրի կատարման երրորդ տակտից՝ EX փուլում գտնվում է համեմատման հրամանը (cmp), որը կատարվում է որպես հանման հրաման, բայց արդյունքը չի պահպանվում: 3-րդ տակտում cmp հրամանը մտնում է կատարման փուլ: beq հրամանը ստուգում է դրոշակների վիճակը կոնվեյերի հաջորդ փուլում՝ EX, այն բանից հետո, երբ cmp-ը կձևավորի դրոշակների բիթերը: MEM փուլում ստուգվում է կանխատեսման ճշտությունը: Թվաբանական և տրամաբանական հրամանները ձևավորում են դրոշակների ռեգիստրի բիթերը: Դրոշակների ռեգիստրը նպատակահարմար է պահել կոնվեյերի կառավարման սարքում:

Դիտարկենք BTB-ի սխեման, որտեղ պահվում են միայն այն անցման հրամանները, որոնք ավարտվել են անցումով: Եթե կատարվել է անպայման անցման հրաման, ապա անցման արդյունքը միշտ կլինի taken: Եթե կատարվել է պայմանական անցման հրաման, ապա անցման հրամանի կատարումից հետո կատարվում է կանխատեսման ճշտության ստուգում: Եթե անցումը կանխատեսվել է սխալ, ապա կոնվեյերը վերաբեռնավորվում է, որի դեպքում ծախսվում են լրացուցիչ տակտեր: Իդեալական դեպքում կոնվեյերի աշխատանքի տակտերի քանակը հավասար կլինի 32-ի: Կառավարման կոնֆլիկտները հաշվի առնելով՝ տակտերի քանակը կավելանա:

Որոշենք տուգանքը անցումների սխալ կանխատեսման դեպքում:

Ենթադրենք՝ ըստ Սմիթի պրեդիկատորի անցումների՝ ճիշտ կանխատեսման հավանականությունը 80% է, BTB-ում հայտնվելու հավանականությունը՝ 90%: Ծրագրի այս հատվածի կատարման դեպքում պայմանական անցումների քանակը հավասար է 11-ի: Անպայման անցումների քանակը՝ 5 է (ճիշտ կանխատեսման հավանականությունը 100%): Եվ քանի որ, ինչպես նշվել է վերը, 5-աստիճան կոնվեյերի դեպքում տույժը կկազմի 3 տակտ, ապա տվյալ ծրագրի կատարման դեպքում տակտերի ընդհանուր քանակը կորոշվի հետևյալ բանաձևով՝ $N=32+(11x0,1+11x0,2)x3=41,9$:

2. Կոնվեյերի տակտերի քանակի գնահատում՝ պայմանական հրամաններով ծրագրի դեպքում: Հրամանների կատարման հաջորդականությունը ցուցադրված է ստորև՝ A=56, B=32, r0=56, r1=32:

1). cmp r0,r1 ; check
2). subgt r0, r0, r1 ; r0= 24; r1=32.
3). sublt r1, r1, r0 ; nop
4). bne gcd ; reach the end? No
5). cmp r0,r1 ; check
6) subgt r0,r0,r1 ; nop
7). sublt r1,r1,r0 ; r0=24, r1=8
8). bne gcd ; end? No
9). cmp r0,r1 ; check
10). subgt r0, r0, r1 ; r0=16; r1=8
11). sublt r1, r1, r0 ; nop
12). bne gcd ; end? No
13). cmp r0,r1 ; check
14). subgt r0, r0, r1 ; r0=8; r1=8
15). sublt r1, r1, r0 ; nop
16). bne gcd ; end? Yes

Յուրաքանչյուր հրամանի կատարման ընթացքում կոնվեյերի վիճակը ներկայացված է աղ. 3-ի մի հատվածը՝ աղ.2-ի նման:

Աղյուսակ 3

	IF	ID	EX	MEM	WB
1.	cmp	-	-	-	-
2.	subgt	Cmp	-	-	-
3.	sublt	Subgt	Cmp	-	-
4.	bne	Sublt	subgt	cmp	-
5.	cmp	Bne	sublt	subgt	Cmp
.	…	…	…	…	…
20.					Bne

Subgt հրամանը հայտնվում է 4-րդ տակտի կատարման ժամանակ: Նույն տակտում bne հրամանը տրվում է կոնվեյերին: Ենթադրենք՝ ro<r1: Subgt հրամանը չի կատարվի, քանի որ դրա կատարման պայմանը չի կատարվում: subgt-ը փոխարինվում է no_operation-ով (nop): Երբ subgt-ը տեղափոխվում է հաջորդ փուլ (MEM), ձևավորվում է, այսպես կոչված, «փուչիկ» (bubble): Հիշողությունից, ինչպես և EX փուլից հետո օպերանդը տեղադրվում է MEM/WB ռեգիստրում: WB

փուլում տեղի է ունենում գրանցում GPR-ներում: Store հրամանի կատարումը տեղի է ունենում հիշողությանը դիմելու համար: WB փուլը բաց է թողնվում [6]:

Կոնվեյերի տակտերի քանակը իդեալական դեպքում հավասար կլինի 20-ի: Հաշվի առնելով կառավարման կոնֆլիկտները՝ տակտերի քանակը մեծանում է: Կանխատեսումը կատարվում է միայն մեկ bne հրամանի դեպքում: Անցումների սխալ կանխատեսման դեպքում սահմանենք տույժ (տուգանք): Ենթադրենք՝ Սմիթի կանխատեսմամբ անցումների ճիշտ կանխատեսելու հավանականությունը 80% է, BTB-ում հայտնվելու հավանականությունը՝ 90%: Պայմանական անցման հրամանների քանակը 4 է: Անպայման անցման հրամաններ չկան: Տվյալ ծրագրի կատարման դեպքում տակտերի ընդհանուր քանակը որոշվում է հետևյալ բանաձևով՝ N=20+(4x0,1+4x0,2)x3=21,2: Պայմանական հրամաններ օգտագործելիս նույն առաջադրանքը լուծելու համար պահանջվում են երկու անգամ քիչ կոնվեյերի տակտեր՝ S=41,9/21,2 ≈1,98

Loop unrolling - կոմպիլյատորի տեխնոլոգիան: Loop unrolling-ը թույլ է տալիս նվազեցնել անցման հրամանների կատարման քանակը: Սա ցիկլի փոխակերպում է, որում ցիկլի մարմնի կոդը բազմիցս կրկնվում է: Ցիկլի կրկնությունների քանակը կրճատվում է (մեկ իտերացիայի ընթացքում կատարվում են միանգամից մի քանիսը):

Օրինակ 1.

```
for (sum = 0, i = 0; i < n; i++) { sum += p[i]; }
loop unrolling –ից հետո
for (sum = 0, i = 0; i < n; i += 4)
{ sum += p[i];
sum += p[i + 1];
sum += p[i + 2];
sum += p[i + 3]; }
```

Իտերացիաների քանակը նվազում է 4 անգամ: Բայց այստեղ կա տվյալների RAW կախվածություն: Ավելի հաջողված տարբերակն է.

```
t1 = t2 = t3 = t4 = 0;
{for (sum = 0, i = 0; i < n; i += 4)
{ t1 += p[i];
t2 += p[i + 1];
t3 += p[i + 2];
t4 += p[i + 3]; }
sum = t1 + t2 + t3 + t4; }
```

Եզրակացություն: Աշխատանքում կիրառվել է անցումների իրականացման 2 մոտեցում՝

• անցումների կանխատեսում,

• պայմանական կատարում կամ պրեդիկացիա:

Անցումների կանխատեսման համար օգտագործվել է Սմիթի պրեդիկտորը, որն ապահովում է ճիշտ կանխատեսման բարձր հավանականություն: Նախընտրելի է այն օգտագործել երկար if-else բլոկների դեպքում:

«Պայմանական հրաման» տերմինը նշանակում է, որ հրամանը կկատարվի ինչ-որ պայմանի բավարարարման դեպքում կամ կանտեսվի՝ նրա չկատարման դեպքում: Պայմանական հրամանները հարմար է օգտագործել կարճ if-else բլոկների կատարման ժամանակ:

• Կոնվեյերում պայմանական հրամանի կատարման ժամանակ հապաղում չկա: Հրամանը պարզապես չի կատարվում:

• Եթե ենթադրենք, որ անցումների մոտավորապես 50%-ը կփոխարինվի պայմանական հրամաններով, ապա սխալ կանխատեսումների թիվը կկրճատվի երկու անգամ: Դա կբարձրացնի կոնվեյերի աշխատանքի արտադրողականությունը և կնվազեցնի էներգիայի սպառումը այն բանի հաշվին, որ իզուր կատարվող հրամանների թիվը կնվազի:

• Փաստորեն, եթե հրամանը չի կատարվում, կորչում է 1 տակտ, բայց չկատարելու պատճառով էներգիայի սպառումը նվազում է: Մշակվել է կոնվեյերների սխեման:

ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ

1. **Цилькер Б.Я**., **Орлов С.А.** Организация ЭВМ и систем. - СПб.: Питер, 2016- 667 с.
2. **Таненбаум Э., Остин Т.** Архитектура компьютера. - 6-е издание. -СПб.: Питер, 2019. -816 с.
3. **Паттерсон Д., Хеннесси Дж.** Архитектура компьютера и проектирование компьютерных систем. –СПб.: Питер, 2012. -784 с.
4. **Hennessy John L., Patterson David A.** Computer Architecture. A Quantitative Approach.- 5-th edition. - 2018.-721 p.
5. **Harris David Money, Harris Sara L**. Digital Design and Computer Architecture. Second edition. - 2019.-1625 p.
6. **Столлингс У.** Структурная организация и архитектура компьютерных систем.- 5-е изд. –М.: Изд.дом “ Ильамс”, 2002.-893 с.

А.К. ТУМАНЯН, А.Г. КАМАЛЯН

ИССЛЕДОВАНИЕ МЕТОДОВ ОРГАНИЗАЦИИ УСЛОВНЫХ ПЕРЕХОДОВ И ПРЕДИКАТНЫХ КОМАНД В МИКРОПРОЦЕССОРНЫХ СИСТЕМАХ

Рассмотрены два способа организации реализации ветвления в программах: с использованием команд перехода и условных (предикатных) команд. Разработана система команд процессора, включающая условные команды. Предложена конвейерная схема условных команд для блока выполнения (Ex) команд.

Ключевые слова: конвейер, схема предсказания, предиктор, предикатные команды, развертывание цикла (Loop unrolling).

A.K. TUMANYAN, A.G. QAMALYAN

RESEARCH OF THE METHODS FOR ORGANIZING CONDITIONAL TRANSITIONS AND PREDICATIVE COMMANDS IN MICROPROCESSOR SYSTEMS

Two ways of organizing the implementation of branching in programs are considered: using transition commands and conditional (predicate) commands. A system of processor commands has been developed, including conditional commands. A pipeline scheme for executing conditional commands for the command execution (EX) block has been developed.

Keywords: pipeline, prediction scheme, predictor, predicate commands, Loop unrolling.

ՀՏԴ 004.383.2

Ծ.Ս. ՀՈՎՀԱՆՆԻՍՅԱՆ, Լ.ՅՈՒ. ԹՈՐՈՍՅԱՆ, Ա.Բ. ԱԹՈՅԱՆ

IoT ՑԱՆՑԵՐՈՒՄ ՄԻՋԱՆԿՅԱԼ ՏՎՅԱԼՆԵՐԻ ԱՐԴՅՈՒՆԱՎԵՏ ՄՇԱԿՄԱՆ ԱՌԱՋԱՐԿՈՒԹՅՈՒՆ

Հետազոտվել են IoT (Internet of Things) ցանցերում տվյալների մշակման և փոխանցման փուլերը: Կատարվել է դրանց համեմատում: Առաջարկվել է IoT ցանցերում եզրային հաշվարկների փուլերում տվյալների մշակման և փոխանցման իրականացման արդյունավետ լուծում՝ հիմնված արհեստական բանականության վրա:

Առանցքային բառեր. IoT ցանց, եզրային հաշվարկներ (edge computing), արհեստական բանականություն (ԱԲ), փոքր տվյալներ (Small Data):

Ներածություն: Այսօր գիտության, տեխնիկայի և տեխնոլոգիական տարբեր բնագավառներում կիրառվում են IoT ցանցերը, որոնց միջոցով հնարավոր է կատարել տարաբնույթ տվյալների հավաքագրում, մշակում և վճիռների կայացում: