MUHAMMAD AL-XORAZMIY NOMIDAGI
TOSHKAENT AXBOROT TEXNOLOGIYALARI
UNIVERSITETI NUKUS FILIALI

KOMPYUTER INJINIRINGI FAKULTETI

Telekommunikatsiya texnologiyalari yo'nalishi 3-bosqich talabasi
Toqsanbaev Qabilbayning
Mikroprotsessor fanidan

MUSTAQIL ISHI

Bajardi: _____
Qabul qildi: _____

Nukus – 2017

Reja:

I. **Kirish**

II. **Asosiy bo’lim**

a) **Mikrokontrollerlarni dasturiy ta’minotini sozlash vositalari**

b) **Operandlarni adreslash usullari**

c) **Mikrokontrollerlarni dasturlash**

d) **Dastur osti dasturini tuzish**

III. **Xulosa**

IV. **Foydalanilgan adabiyotlar**

Kirish

Mikroprotsessorlarning (mikrokontrollerlarning) datchiklar va bajaruvchi qurilmalar bilan o’zaro bog’liqligi periferik qurilmalarning registri (kiritish-chiqarish registri) orqali ma’lumotlarni uzatish yo’li bilan amalga oshadi. Bunday registrlarning alohida joylashgan razryadi periferik qurilmalarning ishlash tartibini, ma’lumotlarni uzatishni tugatishni va shunga o’xshash rejimlarni kiritadi.

Qo’llanilayotgan operandlarni soniga qarab AVR-mikrokontrollerlar komandalarining 3 turi bo’ladi: adressiz, bir adresli va ikki adresli. Birinchi turi da komandalarda faqatgina komanda tomonidan bajarilayotgan funksiyani aniqlaydigan – operatsiyalar kodi (KOP) bor. Ikkinchi va Uchinchi turdagi komandalarda, operatsiyalar kodidan tashqari adreslar qismiga ham ega. Operandaning adresini shakllantirish usulini adresatsiya (addressing) deb ataladi. Adreslash usuli yordamida fizik adres hisoblanadi.

Katta va qiyin dasturda ba’zi bir tugatilgan funktsiyalarni bajaradigan komandalarni ketma-ketligini ajratish mumkin. Agar bunday ketma-ketlikdagi komandalarni alohida modullar – dastur osti dastur (routine, subroutine) ko’rinishida joylashtirilsa, unda, dasturda bu komandalar dastur osti dasturlarni chaqirilish komandalariga almashtirilishi mumkin.

4.7. Mikrokontrollerlarni dasturiy ta’minotini sozlash vositalari.

O’rnatiladigan MP (hamda bir kristalli mikrokontrollerlar) asosidagi qurilmalarning DT ni sozlashni asosiy xususiyati, rivojlangan vositalarning tarkibida yo’qligi hisoblanadi. Ayniqsa o’rnatiladigan mikroprotsessor tizimlari uchun sozlash bosqichi ayni ma’sulyatlidir.

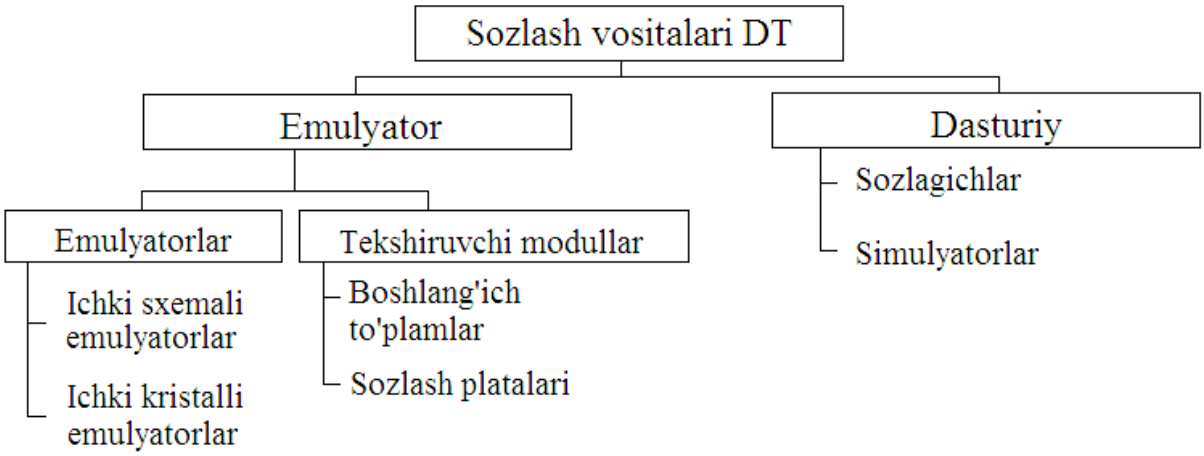
Mikroprotsessorlarning (mikrokontrollerlarning) datchiklar va bajaruvchi qurilmalar bilan o’zaro bog’liqligi periferik qurilmalarning registri (kiritish-chiqarish registri) orqali ma’lumotlarni uzatish yo’li bilan amalga oshadi. Bunday registrlarning alohida joylashgan razryadi periferik qurilmalarning ishlash tartibini, ma’lumotlarni uzatishni tugatishni va shunga o’xshash rejimlarni kiritadi. Bu razryadlarni holati dasturiy o’rnatilishi mumkin. DT sozlashda tez-tez registrlararo uzatish darajasiga o’tib turishga va alohida joylashgan razryadlarni to’g’ri o’rnatilganligini tekshirib turishga to’g’ri keladi. Bundan tashqari, sozlash bosqichida algoritmni optimallashtirish, kodlarni qiyin joylarini topish va ishlab chiqilgan DT ishonchliligini tekshirish mumkin.

Ko’rsatilgan vazifalarni yechish uchun DT sozlash uchun appartli va dasturiy vositalar qo’llaniladi (4.17-chizma).

Sozlashning apparat vositalariga, apparat emulyatori va tekshirish modullari kiradi.

Apparat emulyatori real vaqt rejimida mikroprotsessor tizimlarini dasturiy va apparatli sozlash uchun mo’ljallangan. Ular maxsus DT-sozlash dasturlari bilan ta’minlangan «boshqaruvchi» kompyuter boshqaruvida ishlaydi. Apparat emulyatorlarining asosiy turlari:

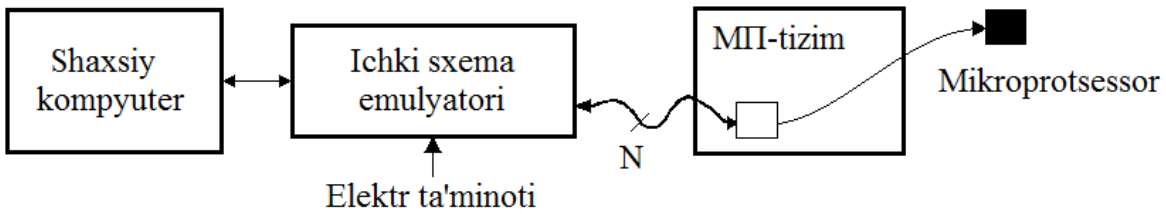
– ichki sxemali emulyatorlar yoki, sozlash tizimida mikroprotsessorlarni o’rnini bosadigan ulanadigan (qo’shiladigan) – emulyatorlar;



4.17-chizma. Dasturiy ta'minotni sozlash vositalarini klassifikatsiyasi

– mikroprotsessorni bitta ichki qurilmalaridan bo'lgan – ichki kristalli emulyatorlar.

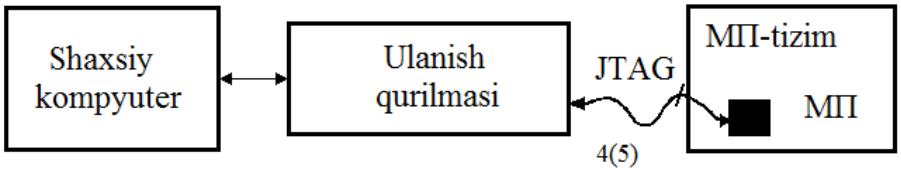
Ichki sxemali emulyator (In-Circuit Emulator, ICE) bu – protsessorning apparatli o'xshatuvchiga (imitator) va o'xshatuvchini boshqarish sxemasiga ega qurilmadir. Emulyator yordami bilan sozlanayotganda mikroprotsessor sozlanayotgan tizimdan chiqarilib olinadi, uni o'rniga kontaktga oid kolodga ulanadi (4.18-chizma). Egiluvchan kabel yordami bilan kontaktga oid kolodga emulyatorga ulanadi. Sozlash jarayonini boshqarish shaxsiy kompyuter orqali amalga oshiriladi. Ulanadigan (qo'shiladigan) - emulyatorlar quyidagi kamchiliklarga ega: qimmat baholigi, kerakli darajada ishonchli emasligi, yuqori energiya iste'moli, emulyator ulangan zanjirlarning elektrik xarakteristikalariga ta'siri.



4.18-chizma. Ichki sxemali emulyator yordami bilan sozlash

Ichki kristalli emulyator (On-Chip Emulator) mikroprotsessorni tizimdan olmasdan turib sozlashga imkon beradi. Bundan tashqari to'g'ridan-to'g'ri dasturni boshqarish mumkin. Ichki kristalli sozlashni eng ko'p tarqalgan vositasi - JTAG (Joint Test Action Group) nomi bilan taniqli, IEEE 1149,1 ketma-ketlik

interfeysidir. JTAG ketma-ketlik sozlash porti, maxsus bog'lanish qurilmalari yordamida kompyuterga ulanadi. Bunda protsessorning sozlash vositalariga ulanishga imkon yaratiladi (4.19-chizma). Bunday sozlash usulini skanlovchi emulyatsiya deb ham atashadi. Bu uslubning ustunligi shundaki, protsessorni tizimdan chiqarmasdan turib har xil ishlarni bajarilish mumkin, tizimning elektrik xarakteristikalarini o'zgartirmagan holda protsessorni maksimal darajada unumdor ishlashini ko'tara olish imkoniyati yaratiladi [14].



4.19-chizma. Ichki kristalli emulyator yordamida sozlash

Tekshiruvchi modullar real vaqt mobavnida dasturiy ta'minotni tez sozlash uchun mo'ljallangan. Tekshiruvchi modullar 2 xil turda bo'ladi: boshlang'ich to'plamlar va sozlash platasi.

Boshlang'ich to'plamlar (Starter Kit) aniq bir mikroprotsessor bilan ishlashni o'rgatish uchun mo'ljallangan. Boshlang'ich to'plam mikroprotsessorni xarakteristikalarini o'rganishga, uncha qiyin bo'lmagan dasturlarni sozlashga, qiyin bo'lmagan maketlashni bajarishga, aniq bir vazifani bajarish uchun mikroprotsessorni qo'llash imkoniyatini tekshirishga imkon beradi. Boshlang'ich to'plam tarkibiga plata, DT va hujjatlar to'plami kiradi. Plataga mikroprotsessor, dasturni yuklaydigan qurilma, ketma-ket yoki parallel portlar, ichki qurilmalar bilan aloqa uchun ajratgichlar va boshqa elementlar o'rnatiladi. Plata kompyuterga parallel yoki ketma-ketli portlar orqali ulanadi. Boshlang'ich to'plamlar mikroprotsessor bilan ishlashni boshlang'ich bosqichida juda ham qulaydirlar.

Sozlash platalari (Evaluation Board) tuzilgan algoritmni real sharoitda tekshirish uchun mo'ljallangan. Ular algoritmni sozlash va optimallashtirish imkonini beradi. Odatda platada mikroprotsessor sinxronizatsiya sxemasi. xotira va periferiya kengayish interfeyslari elektrmanba sxemasi va boshqa qurilmalar

joylashadi. Plata kompyuterga parallel yoki ketma-ketli portlar orqali ulanadi, yoki to’g’ridan-to’g’ri PCI slotiga o’rnatiladi.

Sozlashning asosiy dasturiy vositalari simulyator va sozlagichlardir.

Simulyatorlar (simulator) yoki komandalar tizimini simulyatorlari, protsessorni komandalar darajasida ishini taqlid (imitatsiya) qiladigan dasturdir.

Odatda simulyatorlarni dasturni yoki uni alohida bir qismlarini apparat vositalari da sinashdan oldin tekshirishda qo’llaniladi.

Sozlagichlar (debugger), yaratilgan dasturiy ta’minotni ishlashini tahlil qilishda qo’llaniladigan dasturdir. Sozlagichlarni quyidagi imkoniyatlarini ko’rsatish mumkin.

1. Qadamma-qadam bajarilish. Dastur, komandalar ketma-ketligida, har bir qadamdan keyin boshqaruv sozlagichga qaytish yo’li bilan bajariladi.

2. Haydash. Dasturni bajarilishi ko’rsatilgan komandadan boshlanadi va dastur oxirigacha to’xtovsiz bajariladi.

3. Nazorat nuqtasidan haydash. Dasturni bajarilishi paytida to’xtash sodir bo’ladi va adreslar bilan ishlash komandasi bajarilgandan keyin, ro’yxatda ko’rsatilgan nazorat nuqtalari bilan boshqaruvni sozlagichga beriladi.

4. Registrlar tarkibini va xotira yacheykalarini ko’rish va o’zgartirish. Foydalanuvchi registr tarkibini va xotira yacheykalarini ekranga chiqarish va o‘zgartirish imkoniyatiga ega bo’ladi.

O’rnatiladigan mikroprotsessorlarning DT sozlagichlari odatda ichki sxemali va ichki kristalli emulyatorlar bilan birga ishlatiladi, hamda simulyator rejimida ishlashi mumkin. Ba’zi bir sozlagichlar profillashga, ya’ni dasturni ma’lum bir joyini bajarilganini aniq vaqtini aniqlashga imkon beradi. Ba’zida profillash funksiyasini maxsus dastur - profillagich (profiler) amalga oshiradi.

AVR-mikrokontrollerlarining DT sozlash vositalari, AVR-mikrokontrol-lerlarining dasturiy ta’minotini sozlashni apparat vositalari

ICE50 ichki sxemali emulyatorida JTAG ICE ichki kristalli emulyatorda, hamda STK500 boshlang’ich to’plamida ko’rsatilgan.

AVR Studio muhitining sozlagichi, ICESO ichki sxemali emulyatorida

JTAG ICE ichki kristalli emulyatorda, STK500 boshlang'ich to'plami yoki simulyator bilan qo'llanishi mumkin. Ko'rsatilgan sozlash uslubi loyiha yaratilishi uchun mo'ljallangan. AVR Studio muhitining simulyatori apparat vositalarini qo'llamasdan dasturni dastlabki sozlash uchun mo'ljallangan. AVR Studio muhitida DT sozlash. AVR Studio dasturining sozlash komandalari Debug menyusida joylashgan.

AVR Studio muhitida sozlash rejimiga o'tish Debug menyusidagi Build and Run yoki Start Debugging komandalarini ishlatganda avtomat ravishda ishga tushadi. Sozlash rejimidan chiqish Debug menyusidagi Stop Debugging komandasi orqali amalga oshadi.

Dasturni qadamma-qadam bajarilishi Debug menyusidagi Step Into, Step Over komandalari orqali kiritiladi. Step Into, dasturni bitta komandasini bajarish imkonini beradi (dastur osti dasturni chaqirish komandasini ham).

Dastur osti dasturni bajarilishini tugatish uchun Step Out komandasi ishlatilishi mumkin. Step Over komandasi ham dasturni bitta komandasini bajaradi, lekin agar bu dastur osti dasturni chaqirish komandasi bo'sa, bu komanda bitta qadamni umumiy bajaradi. Keyingi bajariladigan komanda (adresi dasturning schyotchigida joylashgan komanda) dasturning asos matni oynasida ⇨ belgisi bilan belgilanadi. Dasturning bajarilishini uzish Reset komandasi orqali amalga oshiriladi.

Dasturni haydash (bajarishni boshlash va davom ettirish) Run komandasi bilan amalga oshiriladi. Dasturni bajarilishini to'xtatish uchun Break komandasi qo'llaniladi.

Nazorat nuqtalari maxsus markerlardan tashkil topadi va uch xil bo'lishi mumkin: to'xtash nuqtasi, trassirovka nuqtasi va kuzatish nuqtasi.

To'xtash nuqtasi Debug menyusidagi Toggle Breakpoint komandasi yoki dasturning asos matni tahriri menyusidan kiritiladi. Asos matn tahririda to'xtash nuqtasi, belgisi bilan belgilanadi. Kiritilgan to'xtash nuqtalarini Output oynasidagi Breakpoints zakladkasidan ko'rish mumkin Dasturni xuddi o'sha sathda yana to'xtash nuqtasini o'rnitish komandasini qaytadan chaqirish,

to’xtash nuqtasini olinishiga olib keladi. Hamma kiritilgan to’xtash nuqtalarini o’chirish Debug menyusidagi Remove Breakpoints komandasi yoki Output oynasidagi Breakpoints tuguni menyusidagi Remove all Breakpoints komandasi orqali amalga oshiriladi.

To’xtash nuqtasining parametrlari dasturning asos matn tahriri menyusidagi Breakpoints Properties komandasi orqali chaqiriladigan Breakpoint Condition dialog oynasidan kiritiladi. Iterationsni belgilash komandani qayta bajarishlar sonini kiritishga imkon beradi. Watchpoint belgilanganda to’xtash nuqtasigacha yetib kelinishi bilan, faqatgina regjstrlarni qiymati va ko’rish oynalaridagi xotira yacheykalari yangilanishi ishlab chiqiladi. Iterations va Watchpoint larni bir vaqtda belgilash mumkin emas. Show message – to’xtash nuqtasiga yetib kelganligi haqidagi xabarni ko’rsatishi uchun belgilanadi. Xususiyatlarni kiritish va to’xtash nuqtalarini o’chirib tashlash uchun dialog oynasini chaqirish. Output oynasidagi Breakpoints menyusi tugunida amalga oshiriladi.

Trassirovka nuqtasi real vaqt rejimida dasturning bajarilishini boshqarish uchun mo’ljallangan Trassirovka o’zgacha nomga ega – dastur trassasini kuzatishga imkon beradi. AVR Studio muhitida trassirovka funksiyasi faqat ichki sxemali emulyatorni qo’llab, dasturni sozlashda ishlatilishi mumkin. Simulyator rejimi ishlayotganda trasirovka funksiyasini ishlatib bo’lmaydi.

Kuzatish nuqtasi dasturning asos matni tahriri menyusidagi Add to Watch komandasi orqali kiritiladi. Kuzatish nuqtasi, tarkibini kuzatish kerak bo’lgan registr yoki xotira yacheykalarining ramziy nomlarini tashkil qiladi. Add Watch komandasi bajarilayotganda ekranda to’rtta ustunga bo’lingan Watches oynasi paydo bo’ladi. Name (kuzatish nuqtasini ramziy nomi), Value (qiymat), Type (tur), Location (joylashuv).

AVR Studio muhitining sozlagichi yana funksiyalar bilan ta’minlaydi: kursorgacha bajarish (Debug menyusidagi Run to Cursor komanda) va komandalarni ketma-ketlikda bajarish, to’xtash bilan (Debug menyusidagi Auto Step komandasi).

Registrlarni va xotira yacheykalarni tarkibini ko’rish va o’zgartirish uchun View menvusidagi Registers. Memory. Memory I, Memory 2, Memory 3 komandalari xizmat qiladi.

Protsessorni holatini kuzatish uchun Workspace oynasi, I/O zakladkasidagi Processor obyektini ochish lozim. Shunda quyidagi ma’lumot ko’rsatiladi: dastur hisoblagichini tarkibi (Program Counter); stek ko’rsatgichini tarkibi (Stack Pointer); bajarilishni boshidan beri o’tgan taktlar soni (Cycle Counter); 16 razryadli X, Y va Z ko’rsatgich-registrlari tarkibi; takt chastotasi (Frequency), bajarish uchun ketgan vaqt (Stop Watch).

Kiritish-chiqarish registrlarini tarkibini nazorat qilish uchun Workspace oynasi I/O zakladkasidagi I/O* obyektini ochish kerak (* - mikrokontroller turi). I/O obyektiga kiradigan kiritish-chiqarish registrlari, periferik qurilmalar turi bo’yicha guruhlashtirilgan.

Registr va xotira yacheykalari tarkibini o’zgartirilgan (modifikatsiyalangan) qiymati, faqatgina sozlashning joriy seansi vaqtida ishlaydi, dasturning asos matniga o’zgartirishlar kiritilmaydi.

4.8. Operandlarni adreslash usullari

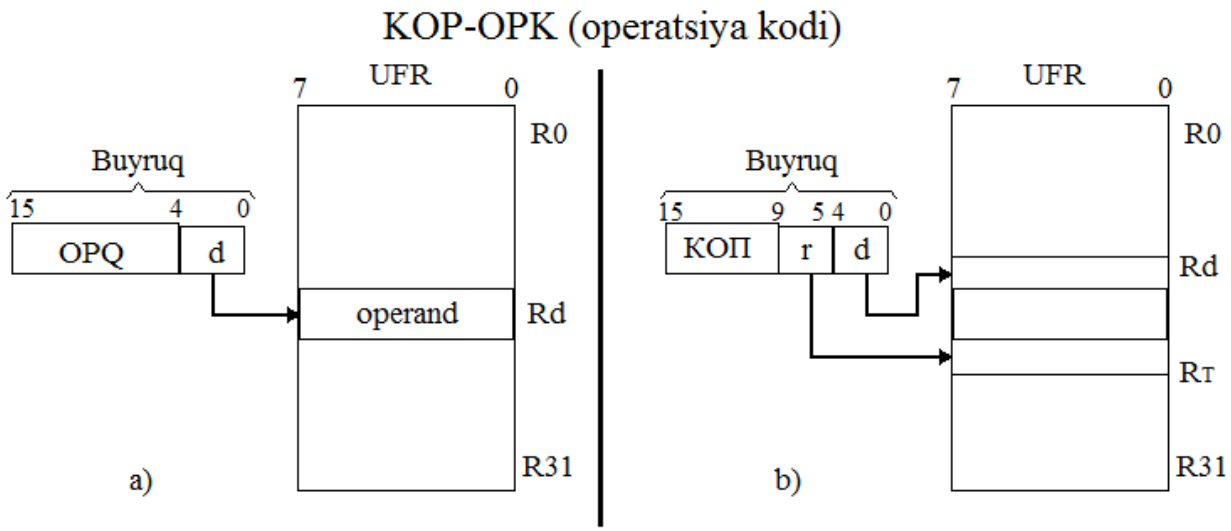
Qo’llanilayotgan operandlarni soniga qarab AVR-mikrokontrollerlar komandalarining 3 turi bo’ladi: adressiz, bir adresli va ikki adresli. Birinchi turi da komandalarda faqatgina komanda tomonidan bajarilayotgan funksiyani aniqlaydigan – operatsiyalar kodi (KOP) bor. Ikkinchi va Uchinchi turdagi komandalarda, operatsiyalar kodidan tashqari adreslar qismiga ham ega. Operandaning adresini shakllantirish usulini adresatsiya (addressing) deb ataladi. Adreslash usuli yordamida fizik adres hisoblanadi.

Xotirani adreslash turiga qarab, AVR–mikrokontrollerda adreslash usullarini – UMR va kiritish-chiqarish registrlarini adreslash usuliga, OXQ adreslash usuliga va dastur xotirasini adreslash usuliga ajratish mumkin. Har xil turdagi adreslash usullarini qo’llanilishi, dasturning hajmini va bajarilishi

uchun ketadigan vaqtim qisqartirish imkonini beradi.

UMR va kiritish-chiqarish registrlarini adreslash uchun faqat bitta rejim ko’zda tutilgan – to’g’ridan-to’g’ri registrli adreslash.

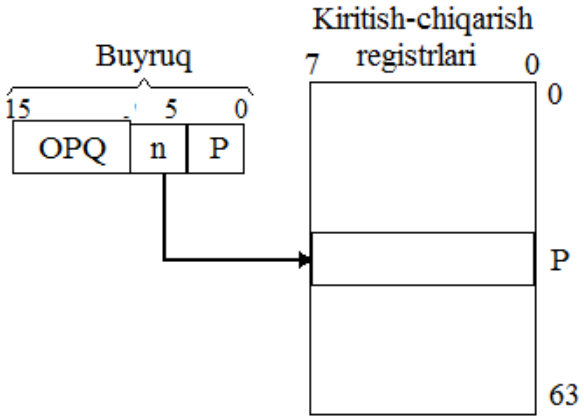
UMRni to’g’ridan-to’g’ri registrli adreslashda, operand – k omandada ko’rsatilgan, UMR tarkibi hisoblanadi. To’g’ridan-to’g’ri registrli adreslash komandalari bitta (Rd) yoki ikkita (Rr va Rd) UMR adreslashligi mumkin (4.20-chizma).



4.20-chizma. To’g’ridan-to’g’ri registrli adreslash

Ikkinchi holatda komandali bajarilishi natijalari Rd registrida saqlanadi. UMR to’g’ridan-to’g’ri registrli adreslash hamma arifmetik va logik komandalarda, hamda bitlar bilan ishlaydigan ba’zi bir komandalarda qo’llaniladi. Chunki bu komandalar ALQda faqat UMR tarkibi ustida bajariladi. Ikkinchi operandasi konstanta (o’zgarmas) bo’lgan komandalarda, birinchi operandasida faqat UMRning katta qismidagi registrlaridan (RI6 – R3I) ishlatilishi mumkin.

Kiritish-chiqarish registrini to’g’ridan-to’g’ri registrli adreslashda, operand - komandada ko’rsatilgan kiritish-chiqarish registirini tarkibida bo’ladi. Kiritish-chiqarish registirini adresi komandaning 6ta razryadida saqlanadi (4.21-chizma).



4.21-chizma. Kiritish-chiqarish registrini to'g'ridan-to'g'ri registrli adreslash.

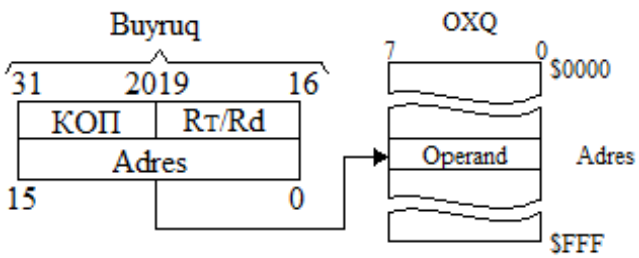
Kiritish-chiqarish registrini to'g'ridan-to'g'ri registrli adreslasha, kiritish-chiqarish registirining IN o'qish va OUT yozish komandalarida, hamda kiritish-chiqarish registri bilan ishlovchi bir qator boshqa komandalarda ishlatiladi.

To'g'ridan-to'g'ri registrli adreslashni qo'llanilishiga misol.

```
                ; bitta UMRni to'g'ridan-to'g'ri registrli adreslash
CLR R1          ; R1 registrini hamma razryadlarini tozalash
                ; ikkita UMRni to'g'ridan-to'g'ri registrli adreslash
ADD R11, R12    ; R11 va R12 registrlarini tarkibini qo'shish
```

Ma'lumotlar operativ xotirasini adreslash uchun 5ta adreslash usuli qo'llaniladi: bevosita, qisman, qisman siljish, qisman predekrementli va qisman postinkrementli.

1. Ma'lumotlar operativ xotirasini bevosita adreslash da, adresi komandada ko'rsatilgan, OXQ yacheykasining tarkibi, operand hisoblanadi. Operandning adresi 32 razryadli komandaning 16 ta kichik razryadida joylashgan bo'ladi (4.22-chizma).



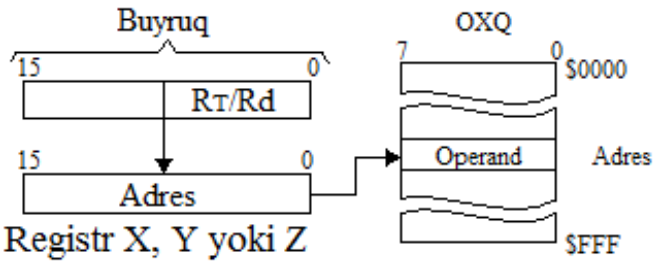
4.22-chizma. Ma'lumotlar operativ xotirasini bevosita adreslash

Bevosita adreslash, LDS (Load Direct from Data Space) komandasida va STS (Store Direct to Data Space) komandasida ishlatiladi. OXQ da baytlarni. byte direktivasi rezervga oladi. byte direktivasi bitta parametrga ega ajratiladigan baytlar soni va faqatgina dseg (data segment) direktivasi yordamida aniqlanadigan ma'lumotlar segmentida ishlatilishi mumkin. Dasturning boshlanish segmenti cseg (code segment) direktivasi yordami bilan ko'rsatiladi. `dseg i. cseg` direktivalari parametrlarga ega emas.

Bevosita adreslashni qo'llanilishiga misol:

```
; bevosita adreslash
dseg              ; ma'lumotlar segmenti (ma'lumotlar operativ xotirasi)
.org $0065        ; $0065 adresi bo'yicha
cntl: .byte1      ; 1 ta baytni cntl uchun rezervlash
cseg              ; dastur segmenti (dastur xotirasi)
;...
LDS R10, cntl     ; cntl ni R10 ga yuklash
```

2. Ma'lumotlar operativ xotirasini qisman adreslashda, adresi X, Y yoki Z registrlarida joylashgan (4.23-chizma), OXQ yacheykasining tarkibi, operand hisoblanadi.



4.23-chizma. Ma'lumotlar operativ xotirasini qisman adreslash

Qisman adreslash LD (Load Indirect) komandasida va ST (Store Indirect) komandasida ishlatiladi. Mi sol uchun:

```
; qisman adreslash
.dseg             ; ma'lumotlar segmenti (ma'lumotlar operativ xotirasi)
cntl: .byte1      ; 1 ta baytni cntl uchun rezervlash
.cseg             ; dastur segmenti (dastur xotirasi)
```

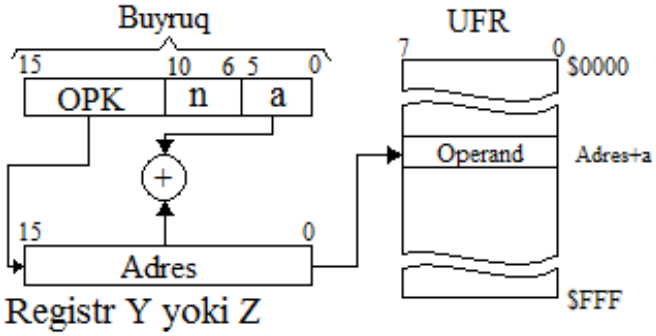
;...

`LDI R30, low(cntl)` ; R30 ga cntl adresining kichik baytini yuklash

`LDI R31, high(cntl)` ; R31 ga cntl adresining katta baytini yuklash

`LD R1,Z` ; R1 ga cntl ni yuklash, ya'ni Rl <-(R31:R30)

3. Ma'lumotlar operativ xotirasini qisman siljishli adreslashda, komandada ko'rsatilgan ma'lumotlar operativ xotirasidagi operanda adresi, Y yoki Z registrlarini tarkibiga siljishni qo'shilish yo'li bilan hisoblanadi. Siljish komandaning 6 ta razryadida joylashadi (4.24-chizma).



4.24-chizma. Ma'lumotlar operativ xotirasini qisman siljishli adreslash.

Qisman siljishli adresatsiya LDD (Load Indirect with Displacement) komandasida va STD (Store Indirect with Displacement) komandasida qo'llaniladi. Misol uchun:

; qisman siljishli adreslash

`.dseg` ; ma'lumotlar segmenti (ma'lumotlar operativ xotirasi)

`arr: .byte 5` ; arr massivi uchun 5ta baytni rezervlash

`.cseg` ; dastur segmenti (dastur xotirasi)

`LDI R30, low(arr)` ; R30 ga arr adresining kichik baytini yuklash

`LDI R31, high(arr)` ; R31 ga arr adresining katta baytini yuklash

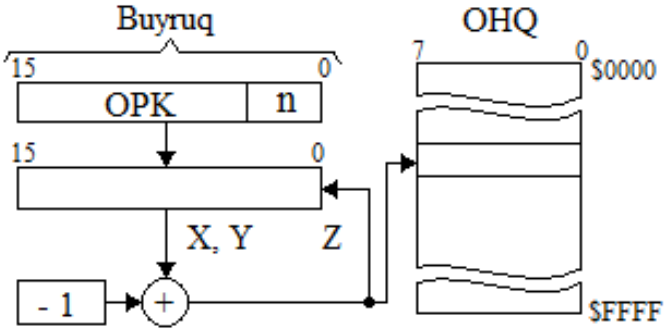
;...

`LDD R10, Z+0` ; arr massivini birinchi elementini R10 ga yuklash

`LDD R11, Z+1` ; arr massivini ikkinchi elementini R11 ga yuklash

4. Ma'lumotlar operativ xotirasini qisman predekrementli adreslashda (lot decrementum – kamayish) tartibi X, Y yoki Z registr komandalarida

ko’rsatilgan komandani bajarilishidan oldin dekrementatsiyalanadi (bitta kamayadi); dekrementatsiyalangan X, Y, yoki Z registrining tarkibi, ma’lumotlar operativ xotirasini operand adresi hisoblanadi (4.25-chizma).



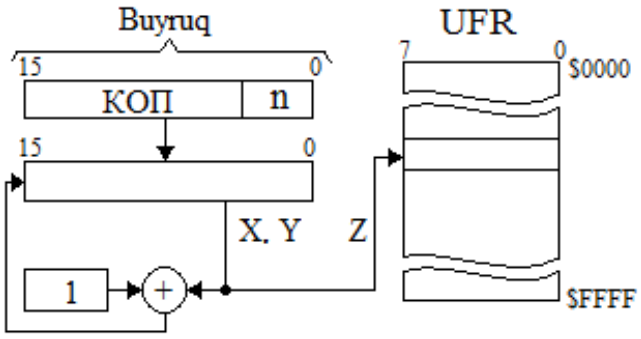
4.25-chizma. Ma’lumotlar operativ xotirasini qisman predekrementli adreslash.

Qisman predekrementli adreslash, LD komandasida va ST komandasida qo’llaniladi. Misol uchun:

; qisman predekrementli adreslash

```
LD    R10, -Z              ; Z<-Z – 1, R10 <-(Z)
```

5. Ma’lumotlar operativ xotirasini qisman postinkrementli adreslash (lot. Incerementum – ko’payish,, o’sish) X, Y yoki Z registrining terkibi ma’lumotlar operativ xotirasini operativ xotirasini operand adresi hisoblanadi; X, Y yoki Z registr tarkibidagi komandalar bajarilganidan keyin inkrementatsiyalanadi, ya’ni bittaga ko’payadi (4.26-chizma).



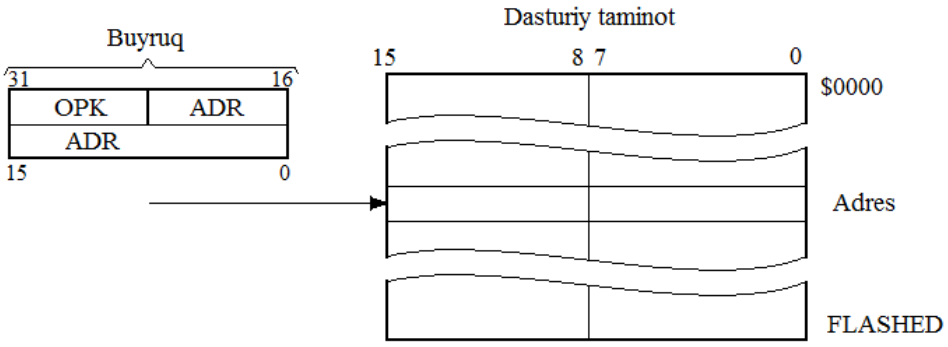
4.26-chizma. Ma’lumotlar operativ xotirasini qisman postinkrementli adreslash

Qisman postinkrementli adreslash, LD komandasida va ST komandasida qo’llaniladi. Misol uchun:

; qisman postinkrementli adreslash

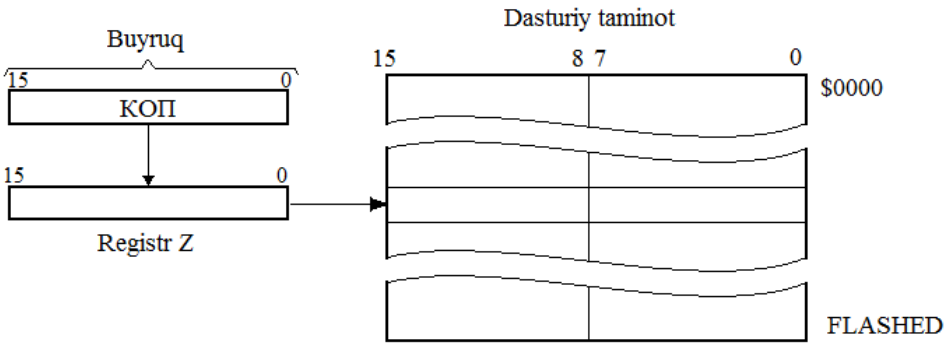
```
LD    R10, Z+              ; R10 <- (Z), Z<-Z+1
```

Dastur xotirasini bevosita adreslashda dasturni bajarilishi, komandada ko’rsatilgan adresdan davom ettiriladi (4.27-chizma). Dastur xotirasini bevosita adreslash JMP va CALL komandalarida qo’llaniladi.



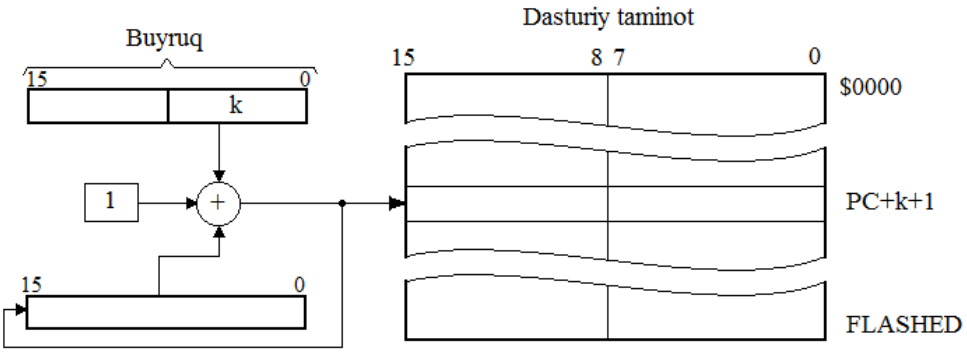
4.27-chizma. Dastur xotirasini bevosita adreslash

Dastur xotirasini qisman adreslashda dasturni bajarilishi, Z registri tarkibidagiadresdan davom ettiriladi, ya’ni dastur hisoblagichiga Z registri tarkibidagi yuklanadi (4.28-chizma). Dastur xotirasini qisman adresatsiyalash IJMP va ICALL komandalarida qo’llaniladi.



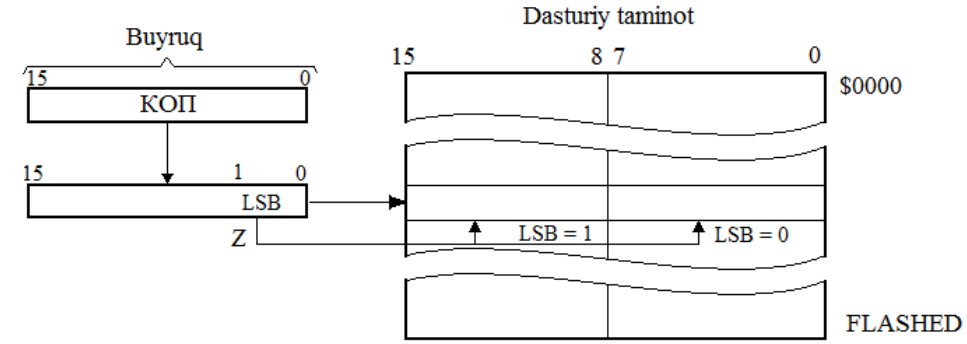
4.28-chizma. Dastur xotirasini qisman adreslash

Dastur xotirasini nisbatan adreslashda dasturni bajarilishi, (PC+k+1) adresidan davom ettiriladi (4.29-chizma). Dastur xotirasini nisbatan adresatsiyalash RJMPi RCALL komandalarida qo’llaniladi.



4.29-chizma. Dastur xotirasini nisbatan adreslash

Konstantani adreslashda konstantaning bayt adresi, Z registrida joylashadi (4.30-chizma). Dastur xotirasida konstantani adresatsiyalsh LPM (Load Program Memory) komandasida qo'llaniladi.



4.30-chizma. Konstantani adreslash

Dastur xotirasiga ma'lumotlarni kiritish. Db (define bytes) direktivasi imkon beradi. Kiritilgan xotira yacheykalariga yo'l ko'rsata olish uchun, direktivadan oldin belgi bo'lishi zarur. Db direktivasi dastur segmentida joylashadi va .org direktivasi bilan qo'llanilishi mumkin.

Dastur xotirasida konstantani adreslashga misol:

```
LDI R31, high(var<<1) ; Z registrining katta bayti
LDI R30, low(var<<1)  ; Z registrining kichik bayti
LPM R16, Z            ; $50 va 137 konstantalar
```

Yuqorida keltirilgan misolda operandani chapga, ko'rsatilgan razryadlar soniga siljitadigan, << operator ishlatilgan.

Dastur xotirasida postinkrementli konstantani adresatsiyalshda konstantaning bayt adresi, Zregistrida joylashadi. Z registridan jo'natilgan bayt

ko’rsatilgan registrga yuklanadi. Z registri tarkibidagi komanda bajarilgandan keyin inkrementatsiyalanadi. Dastur xotirasida postinkrementli konstantani adresatsiyalash LPM va ELPM komandalarida qo’llaniladi. Misol uchun:

```
LDI   R31, high(var<<1)      ; Z registrining katta bayti
LDI   R30, low(var<<1)       ; Z registrining kichik bayti
LPM   R16, Z+      ;$50 sonini R16ga yuklash, inkrement Z
LPM   R17, Z                 ; 137 sonini R17 ga yuklash
;...
var:   .db $50, 137    ; $50 va 137 konstantalar
```

4.9. Mikrokontrollerlarni dasturlash

4.9.1. Siklik dasturlash tuzish

Har qanday boshqarish yoki ma’lumotlarni qayta ishlash jarayoni ba’zi bir algoritmik tuzilishlarni yig’indisini tashkil qiladi. Algoritmik tuzilishlarning eng ko’p tarqalgan turlari – tarmoqlanish (branching) va sikllar (loop) hisoblanadi [15-16].

Tarmoqlanish dasturning har xil qismlarini tuzish (algoritm tarmoqlarini ajratish) uchun qo’llaniladi.

Tsikllarda bitta operatsiya xotira yacheykasida yoki elementlarida ketma-ketlikda joylashgan bir qancha ma’lumotlar tarkibi ustidan bajariladi. Massivlarni va jadvallarni qayta ishlashda sikllidasturlarni qo’llash maqsadga muvofiqdir.

Siklli dasturda 4 ta asosiy blokni ajratish mumkin.

1. Initsiyalizatsiya bloki (lot. Initium – boshlanish). Bunda o’zgaruvchanlarlarga, hisoblagichlarga, indekslarga va ko’rsatkichlarga boshlang’ich qiymatlar kiritiladi.
2. Qayta ishlash bloki. Bunda kerakli hisoblashlar amalga oshiriladi.
3. Sikllarni boshqarish bloki. Bunda keyingi qaytariladigan operatsiyadan oldin hisoblagichlarni va indekslarni (ko’rsatkichlarni) qiymati o’zgartiriladi, hamda sikldan chiqish sharti tekshiriladi.
4. Yakumlovchi blok. Bunda olingan natijalarni saqlash amalga oshiriladi.

2 va 3-bloklar siklni jismini (loop body) tashkil qiladi. Siklik dasturlarni tez ishlash samaradorligini oshirish va hajmini qisqartirish uchun sikl qismini operatsiyadan bo’shatish kerak.

Siklik dasturlarni tashkillashtirish uchun, hamda tarmoqlashtirish uchun dasturlarda shartsiz va shartli komandalar ishlatiladi. Bundan tashqari, sikllarni qurish uchun, bir vaqtda bir qancha ishni bajaradigan, sikllarni maxsus komandalari qo’llanilishi mumkin.

JMP, RJMP, IJMP va EIJMP shartsiz o’tish komandalari boshqaruvni komandada ko’rsatilgan dastur xotirasini adresiga uzatadi. JMP (Jump)

komandasi boshqaruvni dastur xotirasini butun hajmi ichidan uzatishga imkon beradi. RJMP (Relativa Jump) komandasi dastur hisoblagichi joriy tarkibiga nisbatan, ±2 K so'z (±4 Kbayt) me'yorda o'tishlar bilan ta'minlaydi. IJMP (Indirect Jump) komandasi bo'yicha Z registrida ko'rsatilgan adres bo'yicha nisbiy o'tishlarni bajaradi; maksimal siljish 64 K so'z (128 Kbayt)ni tashkil qiladi. EIJMP (Extended Indirect Jump) komandasi dastur xotirasini butun hajmi bo'yicha nisbiy o'tishlar bilan ta'minlaydi; dastur hisoblagichini kengaytirish uchun EIND registri qo'llaniladi.

Shartli o'tishlar komandalari ba'zi bir shartlar bajarilganda dastur xotirasidagi ko'rsatilgan adres bo'yicha boshqaruvni uzatadi.

BR*xx* (Branch if ... – o'tish, agar ...) komandalari, SREG holat registri razryadini tekshiruvi natijalari bo'yicha, dastur hisoblagichining joriy tarkibiga nisbatan, -64...+63 so'z oralig'ida o'tishni bajaradi. SREG holat registri kiritish-chiqarish registrining adres muhitida bo'ladi. Shartlar kodi (C, Z, N, V, S, H) arifmetik, logik komandalar va bitlar bilan ishlash komandalari bajarilganda, holat registrida shakllanadi. Agar natijaning katta razryadidagi siljish komandasi bajarilganda, S (carry – siljish) razryadi o'rnatiladi. Agar komandaning bajarilish natijasi nolga teng bo'lsa, Z (zero – nol) razryadi o'rnatiladi. Agar katta razryad natijasi 1 ga teng bo'lsa, N (negative – salbiy natija) razryadi o'rnatiladi. Agar komandaning bajarilishida belgili sonni razryadli setkasi to'lib qolishi sodir bo'lgan bo'lsa, V (overflow – to'lish) razryadi o'rnatiladi. Belgili sonni razryadli setkasi to'lib qolganda, S=N+V (sign – belgi) belgi natijasini to'g'ri ko'rsatadi. Agar komanda bajarilish paytida natijaning uchinchi razryadidan siljish sodir bo'lgan bo'lsa, H (half carry – yarimsiljish) o'rnatiladi.

Operandlarni solishtirishda tarmoqlanish uchun qo'llaniladigan, shartli o'tishlar komandalari 4.1-jadvalda keltirilgan.

Operandlarni solishtirishda tarmoqlanish uchun qo'llaniladigan, shartli o'tishlar komandalari

4.1-jadval

hart	Logik ifoda	Komanda		Operandlar
		Tekshirish	O'tish	
Rd > Rr	$Z \cdot (N \oplus V) = 0$	CP Rr, Rd	BRLT	Belgili
	$C + Z = 0$	CP Rr, Rd	BRLO	Belgisiz
Rd ≥ Rr	$(N \oplus V) = 0$	CP Rd, Rr	BRGE	Belgili
	$C = 0$	CP Rd, Rr	BRSH/BRCC	Belgisiz
Rd = Rr	$Z = 1$	CP Rd, Rr	BREQ	Belgili Belgisiz
Rd ≠ Rr	$Z = 0$	CP Rd, Rr	BRNE	Belgili Belgisiz
Rd ≤ Rr	$Z + (N \oplus V) = 1$	CP Rr, Rd	BRGE	Belgili
	$C + Z = 1$	CP Rr, Rd	BRSH	Belgisiz
Rd < Rr	$N \oplus V = 1$	CP Rd, Rr	BRLT	Belgili
	$C = 1$	CP Rd, Rr	BRLO/BRCS	Belgisiz

Shartli o'tishlar komandalariga yana CPSE (Compare and Skip if Equal – solishtirish va o'tkazib yuborish, agar teng bo'lsa) komandasi kiradi. Bu komanda ikkita UMR tarkibini solishtiradi va, agar tarkibi bir xil bo'lsa keyingi komandadan keyin o'tkazib yuboradi.

Muvofiq shart bajarilsa, SBRC, SBIS, SBIC (Skip if Bit in Register [I/o Register] is Set [Cleared]) komandalari keyingi komandani o'tkazib yuboradi. Massivlarni qayta ishlashda siklik dasturlarda ma'lumotlar xotirasini predekrementli va postinkrementli nisbiy adreslashni, hamda ma'lumotlar xotirasini siljishli nisbiy adreslashni qo'llash foydali bo'ladi.

4.32-chizmada dasturni bir qismi keltirilgan. Bunda 100 soni, besh baytdan tashkil topgan massiv yacheykasiga kiritiladi. Sikldan chiqish va boshqaruvni uzatish shartlarini uzatish uchun BRNE komandasi ishlatiladi. Siklni qaytarilish chegarasi 5 ga teng, qadam -1 ga teng. Sikl parametri R16 registri tarkibiga kiradi.

```
; ...
array:    .byte 5          ; 5 bayt array massivi uchun
; ...
LDI       R16, 5           ; sikl qaytarilish chegarasi
LDI       R17, 100         ; array massiviga kiritiladigan son
LDI       R18, 1
LDI       R30, low(array)  ; array massiv adresining kichik bayti
LDI       R31, high(array) ; array massiv adresining katta bayti

loop:                      ; sikl jasmi
STZ,          R17          ; array massiviga 100 sonini kiritish
ADD  R30,  R18             ; array massivining keyingi bayti adresi
SUB   R16,  R18            ; o'tishlar soni schyotchigi, qadam -1 teng
BRNE loop                  ; qaytarish, agar schyotchik nolga teng bo'lmasa
; ...
```

4.32-chizma. Massivni siklik qayta ishlash dastur qismi

4.9.2. Dastur osti dasturini tuzish

Katta va qiyin dasturda ba'zi bir tugatilgan funktsiyalarni bajaradigan komandalarni ketma-ketligini ajratish mumkin. Agar bunday ketma-ketlikdagi komandalarni alohida modullar – dastur osti dastur (routine, subroutine) ko'rinishida joylashtirilsa, unda, dasturda bu komandalar dastur osti dasturlarni chaqirilish komandalariga almashtirilishi mumkin. Bunday modulli (tuzilishli) dasturlash quyidagi ustunlikni beradi:

- dasturni sozlash tezlashadi va osonlashadi;
- universal funktsiyalarni amalga oshiradigan dastur osti dasturlar, boshqa dasturlarni tuzishda qo'llanilishi mumkin;
- har xil dastur tillarida yozilgan, translyatsiyalangan dasturdan keyin olingan modullarni, bitta dasturda birlashtirish mumkin.

Chaqirayotgan dasturni dastur osti dastur bilan o'zaro tasiri uchun, quyidagi shartlar bajarilishi lozim;

- chaqirayotgan dasturga, dasturning umumiy tuzilishidagi dastur osti dasturni holati ayon bo'lishi lozim;

- dastur osti dasturni chaqirilish va undan chiqish yo’llari aniqlanishi lozim;

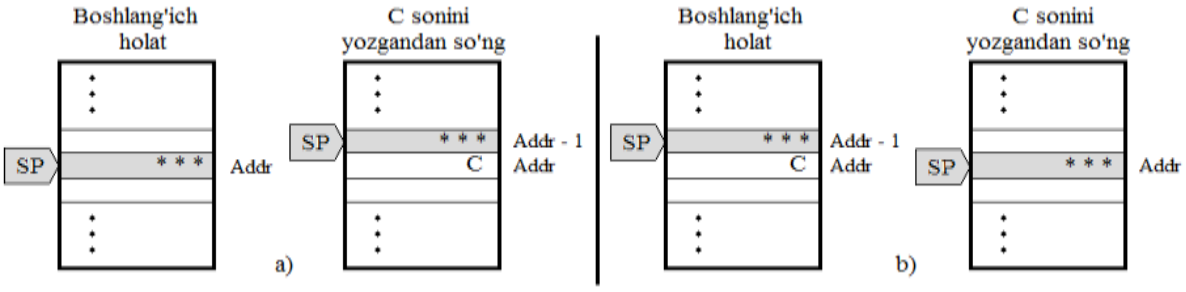
- chaqirayotgan dastur va dastur osti dastur o’rtasida ma’lumotlar almashishni yo’llari tanlanishi lozim.

Dasturning umumiy tuzilishidagi dastur osti dasturning holati, uning nomi bo’yicha aniqlanadi. Dastur osti dastur assemblerdagi nomi, dastur osti dasturni bajarilayotgan qismini boshlang’ich adresi hisoblanadi.

Dastur osti dasturini chaqirish deganda, unga unga komandaning bajarilish qadamini boshqarish tushiniladi. Boshqaruvni topshirish dastur osti dasturni boshlang’ich adresini dastur hisoblagichiga (RS) yuklash yo’li bilan amalga oshiriladi. Qaytarilish adreslarini saqlash uchun dastur osti dasturni *steki* (stack) ishlatiladi.

Navbatdagi bo’sh stekning adresi maxsus registrda – stek ko’rsatkichida SP (Stack Pointer) joylashadi. Son stekga yozilishida yacheykaga adresi bilan joylashtiriladi, bundan keyin stek ko’rsatkichini tarkibi bittaga kamayadi (4.33.a-chizma). Stekdan o’qishda, stek ko’rsatkichining bitta ko’p tarkibli adres bo’yicha tarkibni tanlash amalga oshiriladi (4.33.b-chizma). Shunday qilib, stekka yozishda va undan o’qishda stek ko’rsatkichining tarkibi o’zgaradi.

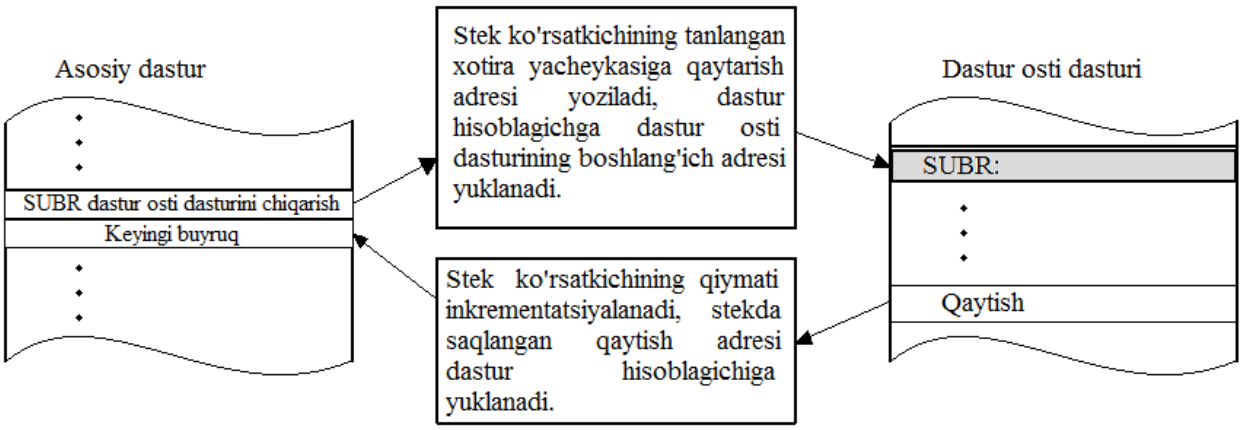
Dastur osti dasturdan qaytarilgan adreslarni saqlash uchun stekni ishlatish mexanizmi, quyidagidan iborat. Qachonki dasturda, dastur osti dasturni chaqirish komandasi uchrasa, stek ko’rsatkichi tanlangan xotira yacheykasiga qaytarish adresi yoziladi, stek ko’rsatkichining qiymati dekrementatsiyalanadi; dastur hisoblagichiga dastur osti dasturning boshlang’ich adresi yuklanadi.



4.33-chizma. Stekka yozish (a) va stekdan o’qish (b) operatsiyasi:

****** - stekning navbatdagi bo’sh yacheykasi; Addr – adres.***

Keyin dastur osti dastur komandasi bajariladi. Dastur osti dasturda qaytarish komandasi uchraganda, stek ko’rsatkichining qiymati inkrementatsiyalanadi; stekda saqlangan qaytish adresi dastur hisoblagichiga yuklanadi (4.34-chizma). Stekni dastur osti dastur bilan ishlatilishi, bitta dastur osti dastur keyingisini chaqirish imkonini beradi, ya’ni dastur osti dasturni joylash imkonini beradi. Dastur osti dasturni joylanish chuqurligi, stek hajmi bilan chegaralandi.



4.34-chizma. Dastur osti dasturni chaqirish va chaqirilgan dasturga qaytish mexanizmi

Ko’p AVR-mikrokontrollerlarda stek operativ xotirada joylashadi. Mikrokontrollerlarda, hajmi 256 baytdan oshmaydigan operativ xotirada stek ko’rsatkichini saqlash uchun faqat bitta SPL (SP) registry qo’llaniladi. Operativ xotirasi bo’lmagan mikrokontrollerlarda uch bosqichli apparat steki bo’ladi.

Ichki operativ xotirada stekni tashkillashtirish quyidagicha bo’lishi mumkin:

```
LDI R16, low(RAMEND)      ; RAMEND adresining kichik qismi
OUT  SPL, R16             ; initsializatsiya SPL
LDI R16, high(RAMEND)     ; RAMEND adresining katta qismi
OUT  SPH, R16             ; initsializatsiya SPH
```

OUT komandasi UMR tarkibini kiritish-chiqarish registriga yozadi; RAMEND – ichki operativ xotiraning so’ngi yacheykasi adresining ramziy nomi Dasturda AVR – mikrokontrollerlarini periferik qurilmalari adresida

ramziy nomlarini ishlatish uchun, `.include` direktivasi yordamida periferik qurilmalarning adresini aniqlash faylini (inc-fayl) qo'shish kerak. Misol uchun, Atmega 8535 mikrokontrolleri uchun m8535def.ing faylini qo'shish lozim:

`.include` "m8535def.inc"

Qo'shilgan inc-faylga AVR Assembler oynasidagi Include Path joyiga yo'l uzatilishi kerak. Bu oynani chaqirish Project menyusidagi AVR Assembler Setup komandasi orqali amalga oshiriladi. `.include` direktivasi qo'llanilayotganda, dasturga .device direktivasini qo'shish shart emas; chunki u inc-faylida joylashgan bo'ladi.

Translyatsiya listingida inc-faylini matnini paydo bo'lishini oldini olishda, listingni o'chiradigan .nolist direktivasi yordam beradi. .nolist direktivasi .list (listingni yoqish) direktivasi bilan birga ishlatiladi. inc-faylini translyatsiya listingidan chiqarish quyidagicha bo'lishi mumkin:

```
.nolist                    ; listing generatsiyasini o'chirish
.include "m8535def.inc"    ; inc-faylini qo'shish
.list                      ; listing generatsiyasini yoqish
```

Assembler tilida dastur osti dasturni chaqirish RCALL, ICALL, CALL va EICALL komandalari orqali amalga oshiriladi. RCALL (Relativa Call – nisbiy chaqiruv) komandasi dastur osti dasturni chaqirishni ta'minaydi (boshlang'ich adresni dastur hisoblagichigini joriy qiymatiga nisbatan siljish – ±2 Kso'z (±4 Kbayt) artofida). ICALL (Indirect Call – qisman chaqiruv) komandasi dastur osti dasturni, dastur xotirasidagi adresdan qisman chaqiruvli ar..alga oshiradi (dastur osti dasturni adresini maksimal siljishi 64 Kso'z (128 Kbayt) ni tashkil etadi). CALL (Call – chaqiruv) komandasi dastur osti dasturni, 4 Mso'z (8Mbayt) gacha hajmga ega dastur xotirasidan chaqirishni ta'minlaydi. EICALL (Extended Indirect Call – kengaytirilgan qisman chaqiruv) dastur osti dasturni, 4 Mso'z (8Mbayt) gacha hajmga ega dastur xotirasidan qisman chaqirishni ta'minlaydi. Ko'rsatkich stekini (SPH:SPL) tarkibi tegishli ravishda 2 yoki 3 taga kamayadi.

Dastur osti dasturni chaqiruv komandasini qo'llash misoli:

```
RCALL subr1              ; subr1 dastur osti dasturni nisbiy chaqiruvi
; subr2 dastur osti dasturni qisman chaqiruv
LDI   R30, low(subr2)
LDI   R31, high(subr2)
ICALL        ; icall komandasi operandalarga ega emas
```

Dastur osti dasturdan qaytish uchun RET komandasi qo'llaniladi. RET komandasi bajarilayotganda, qaytarish adresi stekdan dastur hisoblagichiga yuklanadi.

Stek yana UMR tarkibini dastur osti dasturi bajarilayotgan vaqtida saqlash uchun ham qo'llanilishi mumkin. UMR tarkibini stekda saqlash uchun va chiqarish uchun PUSH va POP komandalari xizmat qiladi. PUSH komandasi registr tarkibidagini, stekka, stek ko'rsatkichida saqlanayotgan adres bo'yicha kiritadi (shunda, stek ko'rsatkichi qiymati bittaga kamayadi (SPH:SPL = SPH:SPL – 1)). POP komandasi teskari ishni bajaradi: stek ko'rsatkichi qiymati bittaga ko'payadi (SPH:SPL = SPH:SPL + 1); stek ko'rsatkichida saqlanilayotgan xotira yacheykasining tarkibi adres bo'yicha registrga yuklanadi.

Dastur osti dastur qo'llanilgan dasturlar, odatda asosiy dasturga nisbatan o'tish komandasidan boshlanadi. Bunda birinchi navbatda stekni initsiyalizatsiya qilish bajariladi (4.35-chizma).

Dastur osti dasturlar bilan ishlanilayotganda, chaqirilayotgan dasturdan dastur osti dasturga parametrlarni uzatishga va dastur osti dasturni bajarilish natijalarini chaqirayotgan dasturga qaytarishga imkon berish muhim ro'l o'ynaydi. Assemblerda chaqirayotgan dastur bilan dastur osti dasturni o'rtasida ma'lumot almashishga imkon berish formatlashtirilmagan. Parametrlarni uzatish uchun UMR, operativ xotira yacheykalari va stek ishlatilishi mumkin.

```
.nolist                          ; listing generatsiyasini o'chirish
.include "m8535def.inc"          ; inc-faylini kiritish
.list                            ; listing generatsiyasini yoqish
RJMP RESET                       ; asosiy dasturga o'tish
PODPR:                           ; PODPR dastur osti dasturi
; …
RET                              ; asosiy dasturga qaytish
RESET:                           ; asosiy dastur
LDI     R16, low(RAMEND)
OUT     SPL, R16                 ; initsiyalizatsiya SPL
LDI     R16, high(RAMEND)
OUT     SPH, R16                 ; initsiyalizatsiya SPH
; …
RCALL PODPR                      ; PODPRdastur osti dasturni chaqirish
; …
```

4.35-chizma. Dastur osti dastur qo'llanilgan dasturga misol

Parametrlarni umumiy maqsadlardagi registr orqali uzatish faqatgina kam sonli parametrlar uchun yariydi. Lekin bu, parametrlarni uzatishda, taxminiy parametrlarga eng tez yo'l beradigan, eng oddiy va ochiq yo'l hisoblanadi.

Parametrlarni operativ xotira orqali uzatish qattiq reglamentar qoidalarni talab qiladi. Misol uchun, uzatilayotgan parametrlarni yoki ularni adresini qiymatlarni to'g'ridan-to'g'ri saqlash uchun xotirada massiv (jadvalni) tashkillashtirish mumkin; boshlang'ich massivni adresini UMR ga kiritish. Boshlang'ich massiv adresiga ega bo'lib, chaqirayotgan dastur va dastur osti kerakli parametrlarga yo'l bo'ladi.

Parametrlarni stek orqali uzatishda dastur osti dasturni chaqirishdan oldin uzatilayotgan parametrlar stekga kiritiladi. Stek ko'rsatkichining qiymati asosiy adres sifatida kiritilganda va ma'lumotlar xotirasini qisman adreslashni qo'llanilganda, stekda joylashgan parametrlarda dasturda yo'l olish mumkin. Misol uchun, agar asosiy dasturda, dastur osti dasturni chaqirishdan oldin, parametrlarning ba'zi bir qiymatlari stekga kiritilsa:

```
LDI    R16, $33          ; R16 <- $33
PUSH   R16               ; R16 registrining tarkibini stekga saqlash unda
```

dasturda unga kirish imkonini olish mumkin:

```
IN     R30, SPL          ; stek ko’rsatkichining katta bayti
IN     R31, SPH          ; stek ko’rsatkichining katta bayti
LDD    R20, Z+3          ; $33 sonini stekdan R20 registriga yuklash
```

IN komandasi kiritish-chiqarish registrini tarkibini UMR ga o’tkazish uchun xizmat qiladi. Shu singari stekni, uzatilayotgan parametrlar massivini adresini saqlash uchun ishlatish mumkin.

Xulosa

Xulosa qilib aytganda o’rnatiladigan MP (hamda bir kristalli mikrokontrollerlar) asosidagi qurilmalarning DT ni sozlashni asosiy xususiyati, rivojlangan vositalarning tarkibida yo’qligi hisoblanadi. Ayniqsa o’rnatiladigan mikroprotsessor tizimlari uchun sozlash bosqichi ayni ma’sulyatli ekan.

Mikroprotsessorlarning (mikrokontrollerlarning) datchiklar va bajaruvchi qurilmalar bilan o’zaro bog’liqligi periferik qurilmalarning registri (kiritish-chiqarish registri) orqali ma’lumotlarni uzatish yo’li bilan amalga oshadi. Bunday registrlarning alohida joylashgan razryadi periferik qurilmalarning ishlash tartibini, ma’lumotlarni uzatishni tugatishni va shunga o’xshash rejimlarni kiritadi.

Qo’llanilayotgan operandlarni soniga qarab AVR-mikrokontrollerlar komandalarining 3 turi bo’ladi: adressiz, bir adresli va ikki adresli. Birinchi turi da komandalarda faqatgina komanda tomonidan bajarilayotgan funksiyani aniqlaydigan – operatsiyalar kodi (KOP) bor. Ikkinchi va Uchinchi turdagi komandalarda, operatsiyalar kodidan tashqari adreslar qismiga ham ega. Operandaning adresini shakllantirish usulini adresatsiya (addressing) deb ataladi. Adreslash usuli yordamida fizik adres hisoblanadi.

Foydalanilgan adabiyotlar

1. **X.YU.Abbasxanova, U.B.Amirsaidov. Mikropeotsessorlar – T.: “Fan va texnologiya”, 2016, 272 bet.**

http://www.ziyonet.uz

http://www.google.ru

http://www.referatlar.uz