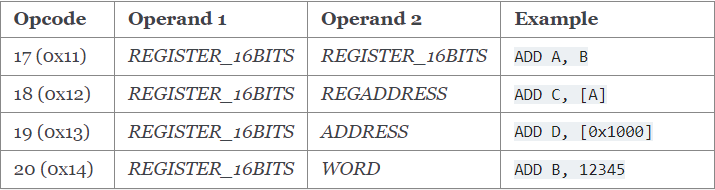
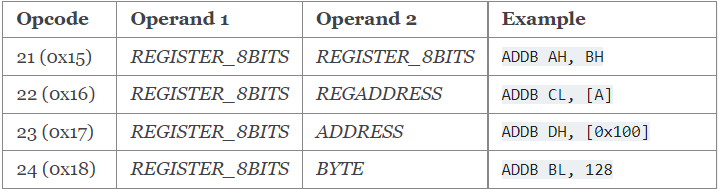
**CPU instruction set**

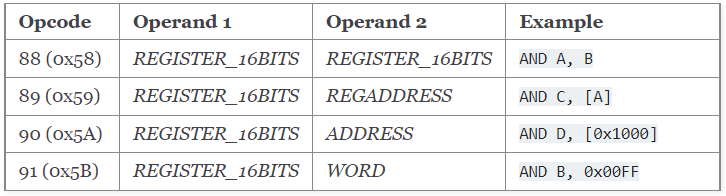
* **ADD: 16-bits addition**
* Vrši sabiranje dva 16-bitna cela broja.
* Svaki oblik instrukcije će imati dva operanda. Operand 1 će uvek biti referenca na 16-bitni registar. Ceo broj sadržan u registru biće dodat vrednosti na koju upućuje operand 2. Rezultat će biti sačuvan u registru na koji upućuje operand 1.



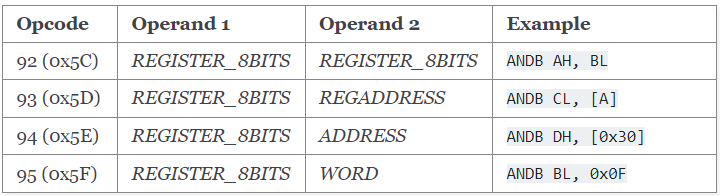
* **ADDB: 8-bits addition**
* Vrši sabiranje dva 8-bitna cela broja.



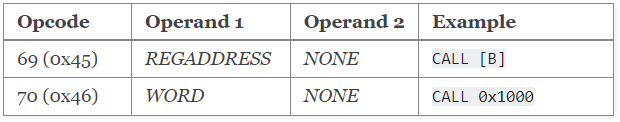
* **AND: 16-bits bitwise AND**
* Izvodi pobitno AND dva 16-bitna cela broja.
* Svaki oblik instrukcije će imati dva operanda.
* Operand 1 će uvek biti referenca na 16-bitni registar.
* Logički AND će biti izveden između sadržaja registra i vrednosti na koju upućuje Operand 2.
* Rezultat će biti sačuvan u registru na koji upućuje Operand 1.



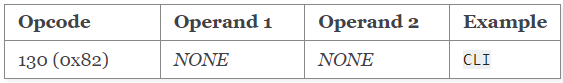
* **ANDB: 8-bits bitwise AND**
* Izvodi pobitno AND dva 8-bitna cela broja.



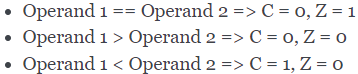
* **CALL: call to subroutine**
* Skoči na potprogram koji počinje na adresi na koju upućuje operand 1.
* Instrukcija će gurnuti u stek povratnu adresu, tj. adresu instrukcije koja sledi nakon poziva.

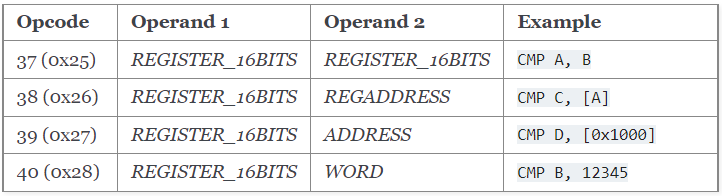


* **CLI: clear interrupt mask**
* Briše bit maske prekida statusnog registra.
* Kada se registar obriše, CPU prekidi su maskirani i, na taj način, onemogućeni.
* Instrukcija nema operande. Ovo je privilegovana instrukcija koja se može pozvati samo kada je u režimu Supervizor.

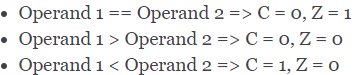


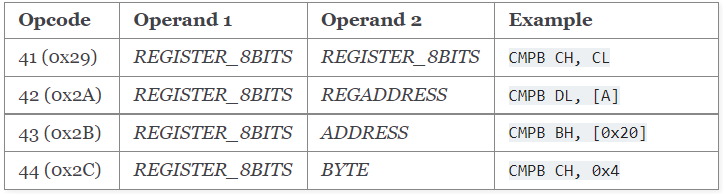
* **CMP: 16-bits integer comparison**
* Upoređuje dva 16-bitna cela broja. Svaki oblik instrukcije će imati dva operanda.
* Operand 1 će uvek biti referenca na 16-bitni registar.
* Poređenje će se obaviti oduzimanjem vrednosti na koju upućuje drugi operand od vrednosti koju sadrži registar na koji upućuje operand 1.
* Rezultat oduzimanja neće biti sačuvan, ali će oznake za prenos (C) i nula (Z) statusnog registra biti modifikovane na sledeći način:





* **CMPB: 8-bits integer comparison**
* Upoređuje dva 8-bitna cela broja.

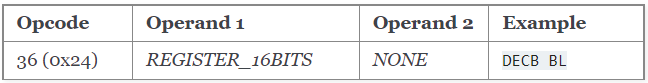




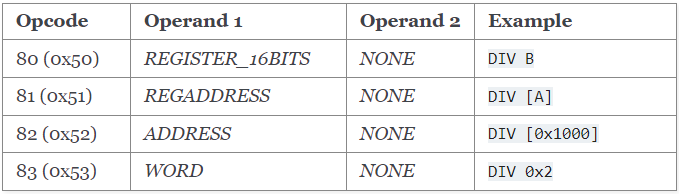
* **DEC: decrement 16-bits register**
* Umanjuje vrednost 16-bitnog registra za 1 jedinicu.
* Rezultat će biti sačuvan u istom registru.
* Operacija će izmeniti vrednosti zastavica za prenos (C) i nula (Z) u registru statusa.



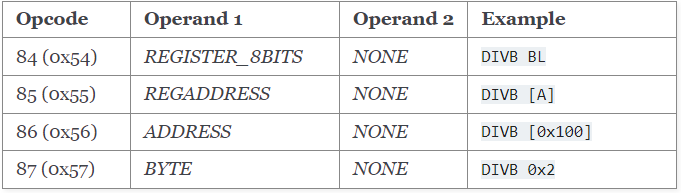
* **DECB: decrement 8-bits register**
* Umanjuje vrednost 8-bitnog registra za 1 jedinicu.



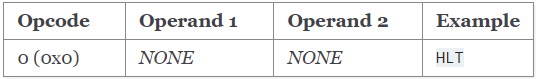
* **DIV: 16-bits division**
* Deli vrednost sačuvanu u registru A sa 16-bitnom vrednošću na koju upućuje operand 1.
* Rezultat će biti sačuvan u Registar A.
* Operacija će izmeniti vrednosti zastavica za prenos (C) i nula (Z) u registru statusa.
* Ako instrukcija izvrši deljenje na nulu, biće pokrenut izuzetak.



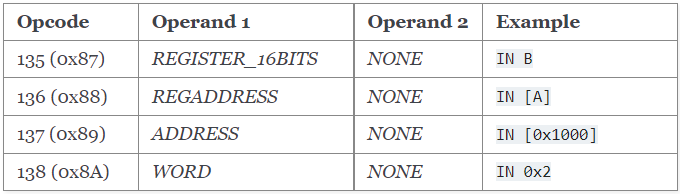
* **DIVB: 8-bits division**
* Deli vrednost sačuvanu u registru A sa 16-bitnom vrednošću na koju upućuje operand 1.



* **HLT: halt processor**
* Postavlja CPU u režim zaustavljanja.
* Zastavica za zaustavljanje (H) statusnog registra će biti postavljena i procesor će biti zaustavljen u izvršavanju daljih instrukcija.
* Do prekida može doći ako su ispravno omogućeni.
* Ako dođe do prekida, CPU će napustiti režim zaustavljanja (zastavica zaustavljanja će biti obrisana) i izvršenje će se nastaviti iz rutine usluge instrukcija.



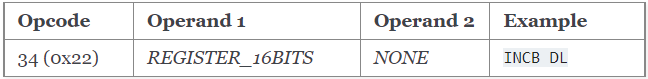
* **IN: read input/output register**
* Čita vrednost ulazno/izlaznog registra.
* Adresa registra koji se čita dobija se iz vrednosti operanda 1.
* Rezultat će biti sačuvan u Registar A. Ovo je privilegovana instrukcija koja se može pozvati samo kada je u režimu Supervizor.



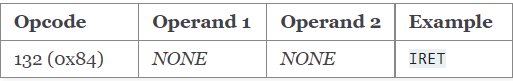
* **INC: increment 16-bits register**
* Povećava vrednost 16-bitnog registra za 1 jedinicu.
* Rezultat će biti sačuvan u istom registru.
* Operacija će izmeniti vrednosti zastavica za prenos (C) i nula (Z) u registru statusa.



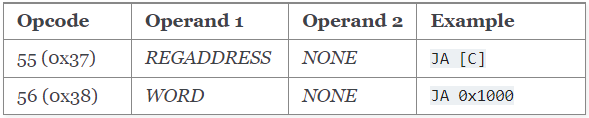
* **INCB: increment 8-bits register**
* Povećava vrednost 8-bitnog registra za 1 jedinicu.

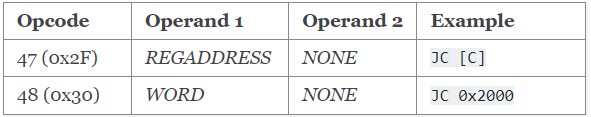
****

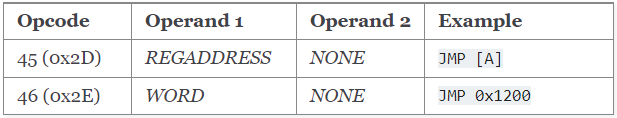
* **IRET: return from ISR**
* Vraća iz rutinera za prekid usluge (ISR).
* Izvršenje ove instrukcije će oporaviti pokazivač instrukcija (IP) i statusni registar uskladištene u steku i skočiti na IP adresu.



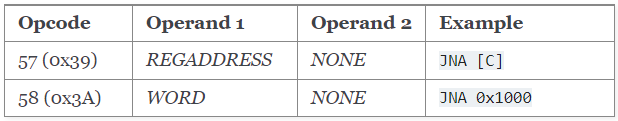
* **JA: jump if above**
* Preskače na datu adresu ako su oznake za prenos (C) i nula (Z) statusnog registra nula (pogledajte CMP: 16-bitno poređenje celog broja).
* Ako je uslov ispunjen, CPU će nastaviti svoje izvršavanje sa adrese na koju upućuje Operand 1.
* U suprotnom, nastaviće se sa sledećim uputstvom. Uputstvo ima jedan mnemonički alias: JNBE.



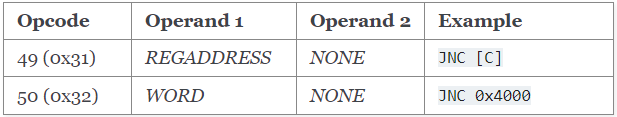
* **JAE: jump if above or equal**
* Pogledaj [JNC: jump if not carry set](https://e.famnit.upr.si/pluginfile.php/662989/mod_resource/content/2/index.html#instruction-jnc).
* **JB: jump if below**
* Pogledaj [JC: jump if carry set](https://e.famnit.upr.si/pluginfile.php/662989/mod_resource/content/2/index.html#instruction-jc).
* **JBE: jump if below or equal**
* Pogledaj [JNA: jump if not above](https://e.famnit.upr.si/pluginfile.php/662989/mod_resource/content/2/index.html#instruction-jna).
* **JC: jump if carry set**
* Skoči na datu adresu ako je postavljena oznaka za nošenje (C) statusnog registra (pogledajte CMP: 16-bitno poređenje celog broja).
* Ako je uslov ispunjen, CPU će nastaviti svoje izvršavanje sa adrese na koju upućuje Operand 1.
* U suprotnom, nastaviće se sa sledećim uputstvom.
* Uputstvo ima dva mnemonička pseudonima: JBE i JNAE.
* **JE: jump if equal**
* Pogledaj [JZ: jump if zero](https://e.famnit.upr.si/pluginfile.php/662989/mod_resource/content/2/index.html#instruction-jz).
* **JMP: jump to address**
* Bezuslovno skače na datu adresu.
* CPU će nastaviti svoje izvršavanje sa adrese na koju upućuje Operand 1.



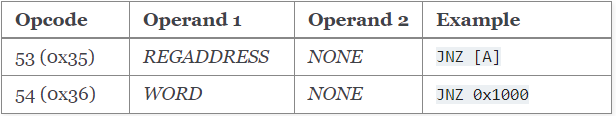
* **JNA: jump if not above**
* Preskače na datu adresu ako su postavljene oznake za prenos (C) ili nula (Z) registra statusa (pogledajte CMP: 16-bitno poređenje celog broja).
* Ako je uslov ispunjen, CPU će nastaviti svoje izvršavanje sa adrese na koju upućuje Operand 1.
* U suprotnom, nastaviće se sa sledećim uputstvom. Instrukcija ima jedan mnemonički alias: JBE.



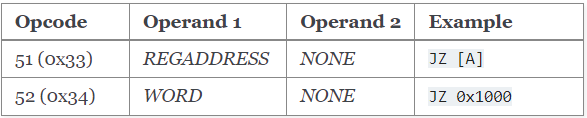
* **JNAE: jump if not above or equal**
* Pogledaj JC: jump if carry set.
* **JNB: jump if not below**
* Pogledaj JNC: jump if not carry set.
* **JNBE: jump if not below or equal**
* Pogledaj JNBE: jump if not below or equal.
* **JNC: jump if not carry set**
* Preskače na datu adresu ako je oznaka za nošenje (C) statusnog registra nula (pogledajte CMP: 16-bitno poređenje celog broja).
* Ako je uslov ispunjen, CPU će nastaviti svoje izvršavanje sa adrese na koju upućuje Operand 1.
* U suprotnom, nastaviće se sa sledećim uputstvom. Uputstvo ima dva mnemonička pseudonima: JNB i JAE.



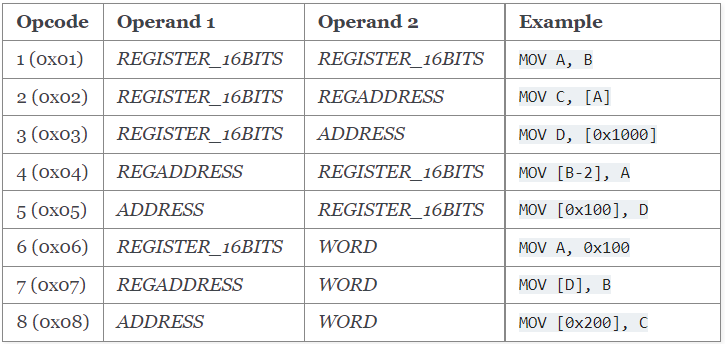
* **JNE: jump if not equal**
* Pogledaj JNZ: jump if not zero.
* **JNZ: jump if not zero**
* Skoči na datu adresu ako je postavljena nula (Z) zastavica statusnog registra (pogledajte CMP: 16-bitno poređenje celog broja).
* Ako je uslov ispunjen, CPU će nastaviti svoje izvršavanje sa adrese na koju upućuje Operand 1.
* U suprotnom, nastaviće se sa sledećim uputstvom.
* Uputstvo ima jedan mnemonički alias: JNE.



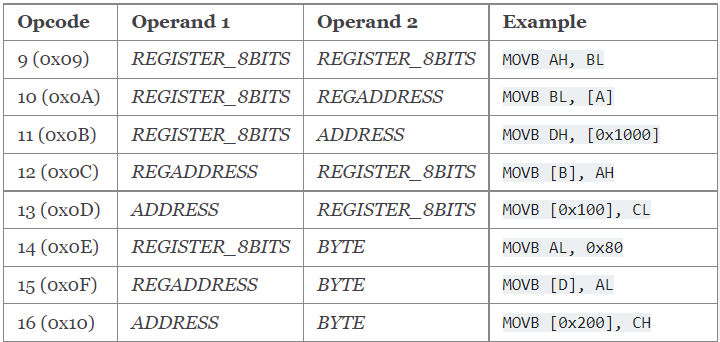
* **JZ: jump if zero**
* Skoči na datu adresu ako je oznaka nula (Z) statusnog registra nula (pogledajte CMP: 16-bitno poređenje celog broja).
* Ako je uslov ispunjen, CPU će nastaviti svoje izvršavanje sa adrese na koju upućuje Operand 1.
* U suprotnom, nastaviće se sa sledećim uputstvom. Uputstvo ima jedan mnemonički alias: JE.



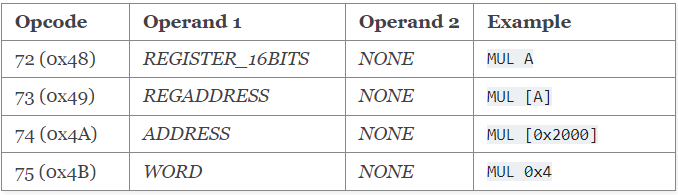
* **MOV: 16-bits copy**
* Kopira 16-bitnu vrednost, na koju upućuje operand 2, na lokaciju na koju upućuje operand 1.



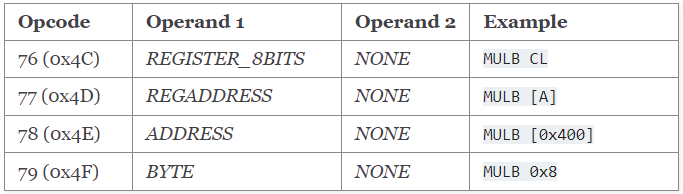
* **MOVB: 8-bits copy**
* Kopira 8-bitnu vrednost, na koju upućuje operand 2, na lokaciju na koju upućuje operand 1.



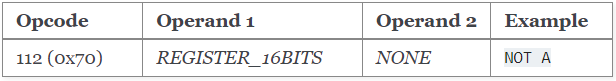
* **MUL: 16-bits multiplication**
* Množi vrednost sačuvanu u registru A sa 16-bitnom vrednošću na koju upućuje operand 1. Rezultat će biti sačuvan u registru A.
* Operacija će izmeniti vrednosti zastavica za prenos (C) i nula (Z) u registru statusa.



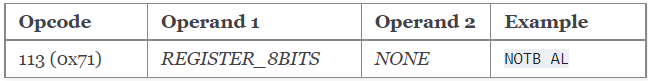
* **MULB: 8-bits multiplication**
* Množi vrednost sačuvanu u registru A sa 8-bitnom vrednošću na koju upućuje operand 1.



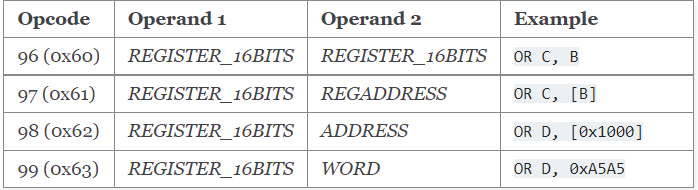
* **NOT: 16-bits bitwise NOT**
* Izvodi NOT u bitovima na bitovima 16-bitnog registra, na koje upućuje operand 1.
* Rezultat operacije biće sačuvan u istom registru.



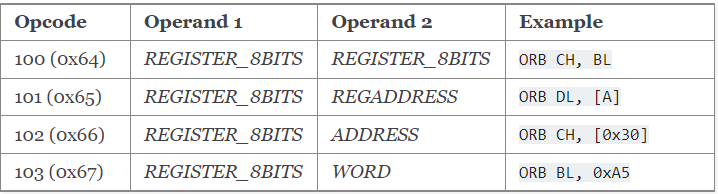
* **NOTB: 8-bits bitwise NOT**
* Izvodi NOT u bitovima na bitovima 16-bitnog registra, na koje upućuje operand 1.



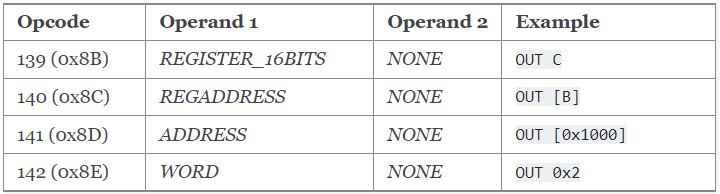
* **OR: 16-bits bitwise OR**
* Izvodi pobitno OR od dva 16-bitna cela broja. Svaki oblik instrukcije će imati dva operanda.
* Operand 1 će uvek biti referenca na 16-bitni registar.
* Logički OR će biti izveden između sadržaja registra i vrednosti na koju upućuje Operand 2.
* Rezultat će biti sačuvan u registru na koji upućuje Operand 1.



* **ORB: 8-bits bitwise OR**
* Izvodi pobitno OR od dva 8-bitna cela broja.



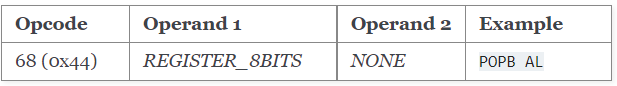
* **OUT: write input/output register**
* Upisuje sadržaj registra opšte namene A u ulazno/izlazni registar.
* Adresa registra koji se upisuje dobija se iz vrednosti operanda 1.
* Ovo je privilegovana instrukcija koja se može pozvati samo kada je u režimu Supervizor.



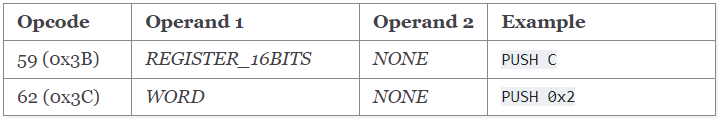
* **POP: pop 16-bits from stack**
* Izbacuje 16-bitnu vrednost sa vrha steka i skladišti je u 16-bitni registar na koji upućuje Operand 1.
* Instrukcija će ažurirati pokazivač steka (SP), povećavajući ga za 2 jedinice.



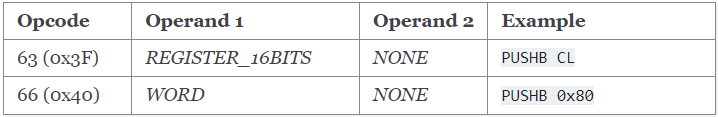
* **POPB: pop 8-bits from stack**
* Izbacuje 8-bitnu vrednost sa vrha steka i skladišti je u 8-bitni registar na koji upućuje Operand 1.
* Instrukcija će ažurirati pokazivač steka (SP), povećavajući ga za 1 jedinicu.



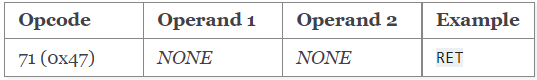
* **PUSH: push 16-bits to stack**
* Gura 16-bitnu vrednost, na koju upućuje operand 1, na vrh steka.
* Instrukcija će ažurirati pokazivač steka (SP), smanjujući ga za 2 jedinice.



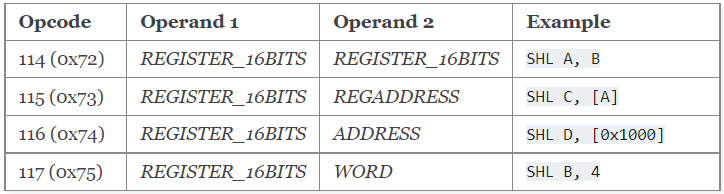
* **PUSHB: push 8-bits to stack**
* Gura 8-bitnu vrednost, na koju upućuje operand 1, na vrh steka.
* Instrukcija će ažurirati pokazivač steka (SP), smanjujući ga za 1 jedinicu.



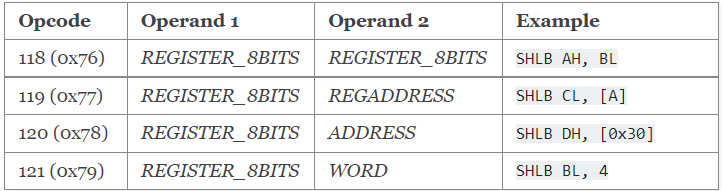
* **RET: return from subroutine**
* Vraća iz potprograma.
* Izvršenje ove instrukcije će pokrenuti pokazivač instrukcija (IP) sačuvan u steku i skočiti na IP adresu.
* Instrukcija će ažurirati pokazivač steka (SP).



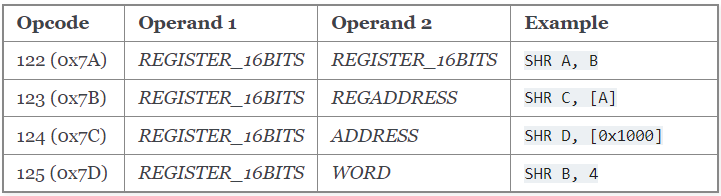
* **SHL: 16-bits logical left shift**
* Izvodi logički levi pomak vrednosti 16-bitnog registra.
* Svaki oblik instrukcije će imati dva operanda.
* Operand 1 će uvek biti referenca na 16-bitni registar.
* Operand 2 će ukazati na broj pozicija bitova na koje će vrednost biti pomerena.
* Rezultat će biti sačuvan u registru na koji upućuje Operand 1.



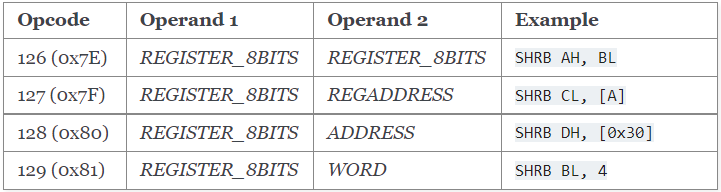
* **SHLB: 8-bits logical left shift**
* Izvodi logički levi pomak vrednosti 8-bitnog registra.
* Svaki oblik instrukcije će imati dva operanda.
* Operand 1 će uvek biti referenca na 8-bitni registar.



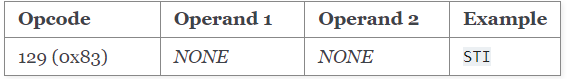
* **SHR: 16-bits logical right shift**
* Izvodi logičko pomeranje udesno vrednosti 16-bitnog registra.
* Svaki oblik instrukcije će imati dva operanda.
* Operand 1 će uvek biti referenca na 16-bitni registar.
* Operand 2 će ukazati na broj pozicija bitova na koje će vrednost biti pomerena.
* Rezultat će biti sačuvan u registru na koji upućuje Operand 1.



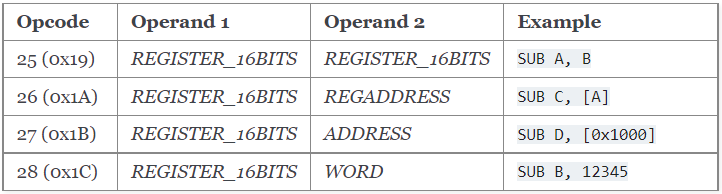
* **SHRB: 8-bits logical right shift**
* Izvodi logičko pomeranje udesno vrednosti 8-bitnog registra.



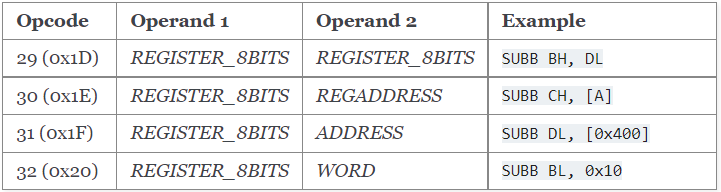
* **STI: set interrupt mask**
* Postavite bit maske prekida statusnog registra.
* Kada se registar obriše, CPU prekidi se demaskiraju i, na taj način, omogućavaju.
* Instrukcija nema operande.
* Ovo je privilegovana instrukcija koja se može pozvati samo kada je u režimu Supervizor.



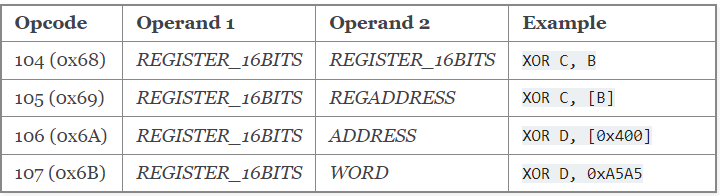
* **SUB: 16-bits substraction**
* Izvodi oduzimanje dva 16-bitna cela broja.
* Svaki oblik instrukcije će imati dva operanda.
* Operand 1 će uvek biti referenca na 16-bitni registar.
* Ceo broj sadržan u registru biće oduzet od vrednosti na koju upućuje operand 2.
* Rezultat će biti sačuvan u registru na koji upućuje Operand 1.



* **SUBB: 8-bits substraction**
* Izvodi oduzimanje dva 8-bitna cela broja.



* **XOR: 16-bits bitwise XOR**
* Izvodi pobitni XOR od dva 16-bitna cela broja.
* Svaki oblik instrukcije će imati dva operanda.
* Operand 1 će uvek biti referenca na 16-bitni registar.
* Logički XOR će biti izveden između sadržaja registra i vrednosti na koju upućuje Operand 2.
* Rezultat će biti sačuvan u registru na koji upućuje Operand 1.



* **XORB: 8-bits bitwise XOR**
* Izvodi pobitni XOR od dva 16-bitna cela broja.

