CLR (Clear Register) : Yazmac icerisinde olan bitlerin hepsini silmek icin kullanilir.

CLR Rd 0 ≤ d ≤ 31

LDI (Load Immediate) : 8 bitlik sabitleri direkt olarak on altinci yazmactan otuz birinci yazmactan herhangi birine yuklemek icin kullanilir.

LDI Rd , K 16 ≤ d ≤ 31 , 0 ≤ K ≤ 255

OUT (Store Register to I/O Location) : Kayıt dosyası icerisindeki yazmaclari giris cikis alanlarina (portlar , zamanlayicilar , konfigurasyon yazmaclari gibi) depolamak icin kullanilir.

OUT A , Rr 0 ≤ r ≤ 31 , 0 ≤ A ≤ 63

IN (Load an I/O Location to Register) : Giris cikis alanlarindan veri almak icin kullanilir.

IN A , Rr 0 ≤ r ≤ 31 , 0 ≤ A ≤ 63

NOP (No Operation) : Tek dongu boyunca hic bir sey yaptirmamak icin kullanilir. Genellikle geri donus degiskeni olmayan labellar icerisine yazilir.

ADD (Add Without Carry) : Iki yazmaci birbiriyle toplamak icin kullanilir.

ADD Rd , Rr 0 ≤ d ≤ 31 , 0 ≤ r ≤ 31 Rd yazmacina Rr yazmacini ekledi.

INC (Increment) veya DEC (Decrement) : Yazmaclara bir eklemek veya cikarmak icin kullanilir

INC Rd ; DEC Rd 0 ≤ d ≤ 31

RJMP (Relative Jump) veya JMP (Jump) : Jump ve RJMP , belirtilen bellekteki adrese gitmek icin kullanilir. Bir nevi labela yani fonksiyona gitmek icin kullanilir. Ikisinin arasindaki fark JMP , daha kapsamli bellekteki adreslere gidebilme kapasitesine sahiptir.

JMP k ; RJMP m 0 ≤ k ≤ 4M -2K ≤ r ≤ 2K

BRNE (Branch If Not Equal) veya BREQ (Branch If Equal) : Bu komuttan once yazilan komut eger istenen kosula esit veya degilse bu yazilan komutta istenen labela yani bir nevi fonksiyona gider.

CP (Compare) : Iki yazmaci karsilastirmak icin kullaniriz. Bu komuttan sonra branch komutlar (BRNE , BREQ gibi) kullanilir.

CP Rd , Rr 0 ≤ d ≤ 31 , 0 ≤ r ≤ 31

AND , OR : Cogu lojik operatorleri de komut olarak kullanabiliriz. Mesela bir degiskenin cift veya tek oldugunu AND operatoru ile anlayabiliriz.

AND Rd , Rr ; OR Rr , Rr 0 ≤ d ≤ 31 , 0 ≤ r ≤ 31

MOV (Copy Register) : Bu komut , bir yazmaci diger yazmaca kopyalamak icin kullanilir.

MOV Rd, Rr 0 ≤ d ≤ 31 , 0 ≤ r ≤ 31

RET (Return from Subroutine) : Bu komutun yazildigi labeldan once cagirilan labela gider. Mesela bir fonksiyona gidilmesi istenmistir ve bu fonksiyonun sonunda ise cagirildigi komut nereye yazilmissa o fonksiyona gider.

SBIC (Skip If Bit in I/O Register Is Cleared) veya SBIS (Skip If Bit in I/O Register Is Set) : Bu komut giris cikis yazmaci icerisindeki tek bir biti test eder ve eger bit temizlenmis veya set edilmis ise bir sonraki komuta gider. Mesela basili butonu control etmek icin SBIC kullanabilirsiniz.

SBIC A,b ; SBIS A,b 0 ≤ A ≤ 31 , 0 ≤ b ≤ 7

PORT ile DDR arasindaki fark : OUT veya IN komutlarindan sonra yazilan bu komutlar aslinda yazilan giris/cikislara anot ve katot ozellikleri eklemek icin kullanilir. PORT , anot gibi davranmasini saglarken ; DDR ise katot gibi davranmasini saglar. Anot ise devrenin tamamlamasi icin topraga baglarken katot icinse devre uzerinde gelen gerilime baglanir.

OUT PORTB , 0b000001 Mesela Arduino UNO uzerinde B kismindan (8-13 portu) sekizinci port uzerinden anot alinabilir demektir. Bir diger ucu ise topraga baglanir.

OUT DDRB , 0b000001 B kismindan (8-13 portu) sekizinci port uzerinden katot alinabilir demektir. Bir diger ucu ise devre uzerinden gelen gerilime baglanir.t

RJMP veya JMP ile CALL arasindaki fark nedir ?

JMP tam olarak yazilmis olan hafiza adresine gider. CALL komutunun da gorevi neredeyse aynidir fakat arasinda bir fark vardir. CALL komutu calismadan once bulundugu komutu saklar ki bu JMP veya RJMP komutuna gore daha efektiftir. Amac , JMP komutunun program boyunca ilerlemesi iken CALL komutu ise yürütmeden sonra, çağrıldığı yerden bir adres sonraki noktaya geri dönen bir alt rutini başlatır.