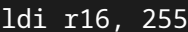


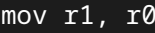
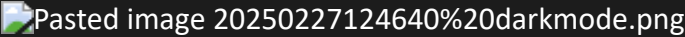
Система команд ATmega8515

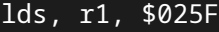
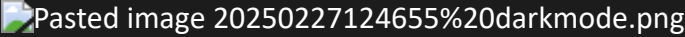
- **Пересылки**
 - Регистр-память - LD, ST, LPM, SPM
 - Регистр-регистр - MOV, IN, OUT
- **Арифметические/логические операции**
 - Сложение, вычитание - ADD, ADDC, SUB, SUBC
 - Умножение - MUL, MULS, etc
 - ДЕЛЕНИЯ НЕТ, только внешние реализации
 - Побитовые и, или, xor - AND, OR, XOR
 - Инкременты/декременты - INC, DEC
- **Передача управления**
 - Безусловные переходы - JMP, RJMP, IJMP, CALL, RCALL
 - Условные переходы
 - Проверка слова состояния - BRNE, BRTC, etc
 - Пропуск следующей команды по условию - SBIC, SBRC, etc
- **Работа с битами**
 - Логический/арифметический/циклический сдвиги - LSR, ASR, ROR, etc
 - Установка битов регистров/флагов - SBI, CBI, SEC, CLC, etc
- **Управление МК** - NOP, SLEEP, WDR

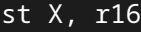
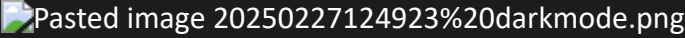
Способы адресации операндов

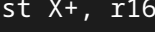
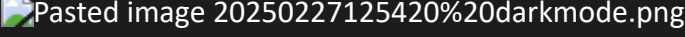
- Непосредственная - константа в команде

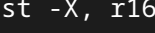
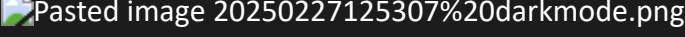

- Прямая регистровая - операнд в регистре, указан в команде

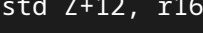
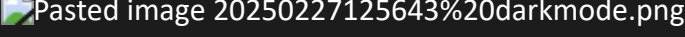


- Прямая - операнд в памяти, в команде указан адрес


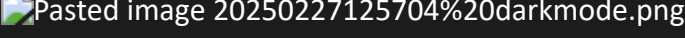


- Косвенная - операнд в памяти данных, адрес в регистрах X-Z


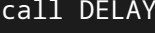
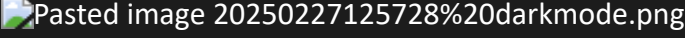


- Косвенная с постинкрементом - операнд в памяти данных, адрес в регистрах X-Z, адрес в X-Z инкрементируется после операции

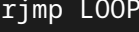
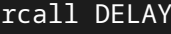
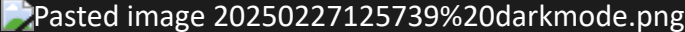


- Косвенная с преддекрементом - операнд в памяти данных, адрес в регистрах X-Z, декремент перед операцией



- Косвенная со смещением - операнд в памяти данных, адрес получается из значения Y/Z + смещение из команды



- Константная адресация памяти программ - адрес байта памяти в регистре Z



- Прямая адресация памяти программ - адрес ячейки памяти программ указан в команде

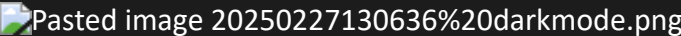



- Относительная адресация памяти программ - адрес ячейки получается через сумму содержимого PC и смещения из команды

Прерывания

Прерывание - передача управления подпрограмме (обработчику прерывания) при поступлении от некоторого устройства сигнала, требующего немедленной обработки
По расположению устройства, посылающего прерывание:

- Внутренние - внутреннее устройство МК: таймер, компаратор
 - Внешние - устройства вне МК: кнопки, датчики, что угодно ещё
- Идентификация источника** прерывания:
- Программная - обработчик в процессе осознаёт кто отправил прерывание
 - Векторная - каждому источнику - отдельный вектор прерывания и собственно свой обработчик
- Таблица векторов прерываний AVR** - в начале Flash памяти



-
1. Микроприкол - отличие JMP от RJMP:
JMP - Jump - умеет прыгать внутри всей области адресации программы (~4 миллиона слов) и доступна не на всех AVR.
RJMP - Relative Jump - прыгает относительно нынешней инструкции в пределах 4 тысяч слов и, судя по всему, доступен везде (адреса в пределах PC - 2K + 1 -- PC + 2K слов)
С CALL и RCALL та же ебень что и с JMP и RJMP
Макроприкол - IJMP - Indirect Jump - прыжок по адресу, лежащему в регистре Z
 2. NOP - буквально нихера не делать (No OPeration)
SLEEP - очевидно - прождать чутка
WDR - ресет сторожевого таймера (WatchDog timer Reset)
 3. Вектор прерывания - адрес обработчика