Budapesti Műszaki és Gazdaságtudományi Egyetem

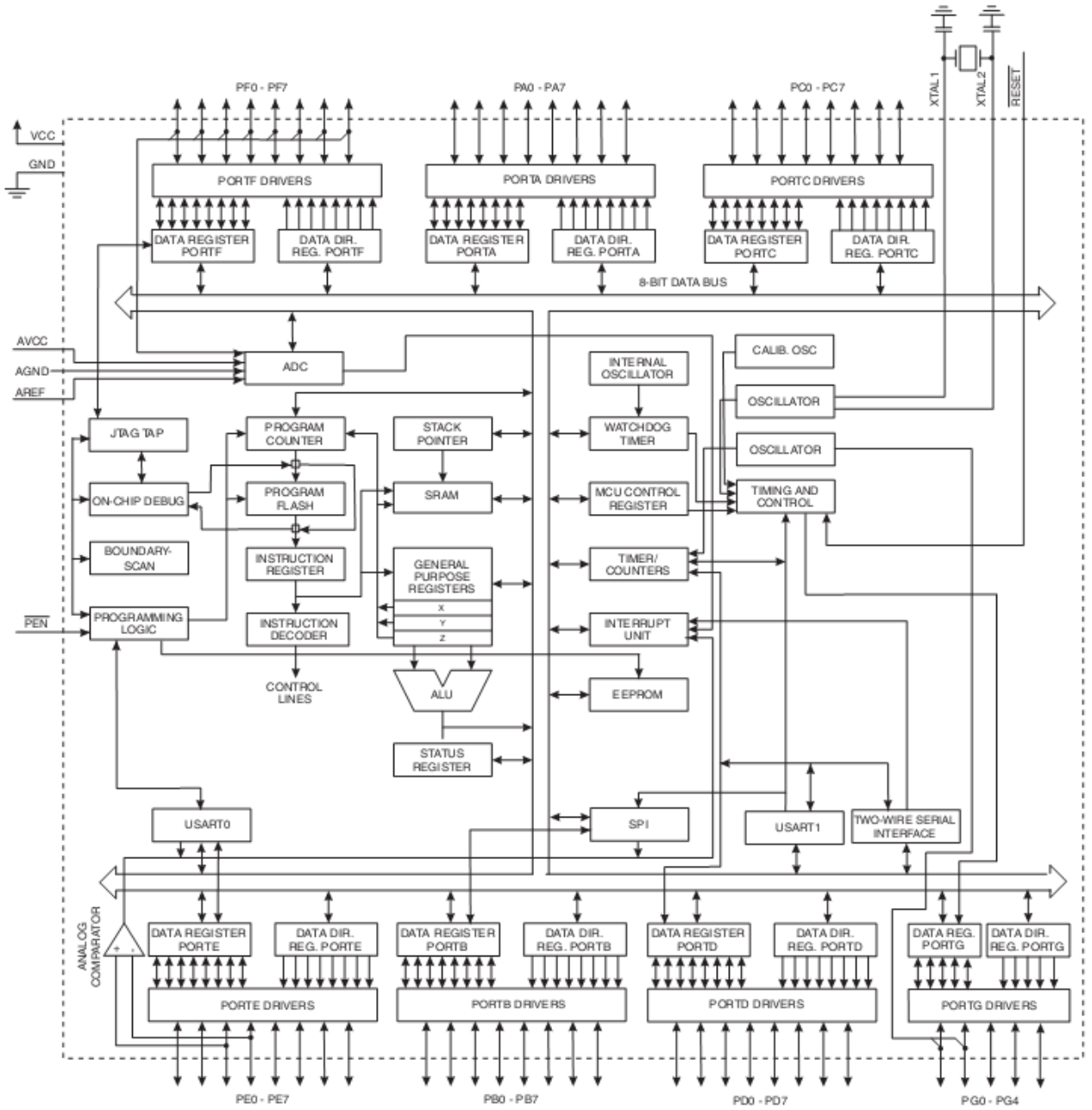
Közlekedésautomatikai Tanszék

Mechatronics and Microcomputers

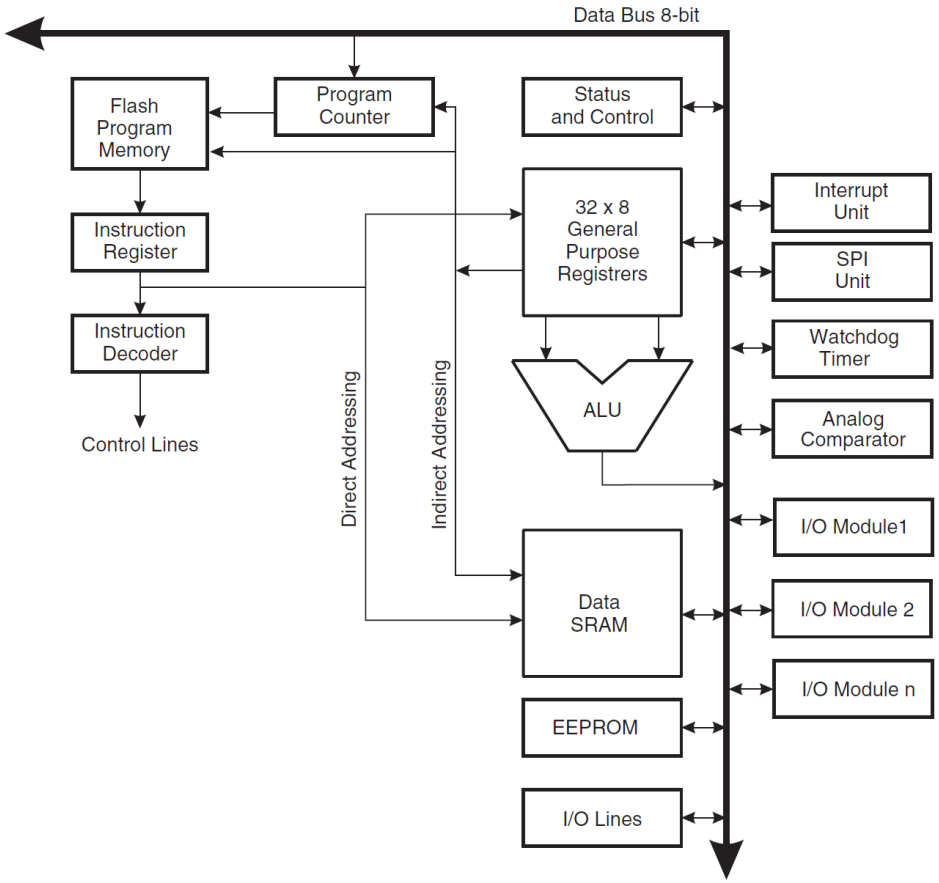
Stipendium Hungaricum

2018/2019 Autumn Semester

Szilárd Aradi, PhD

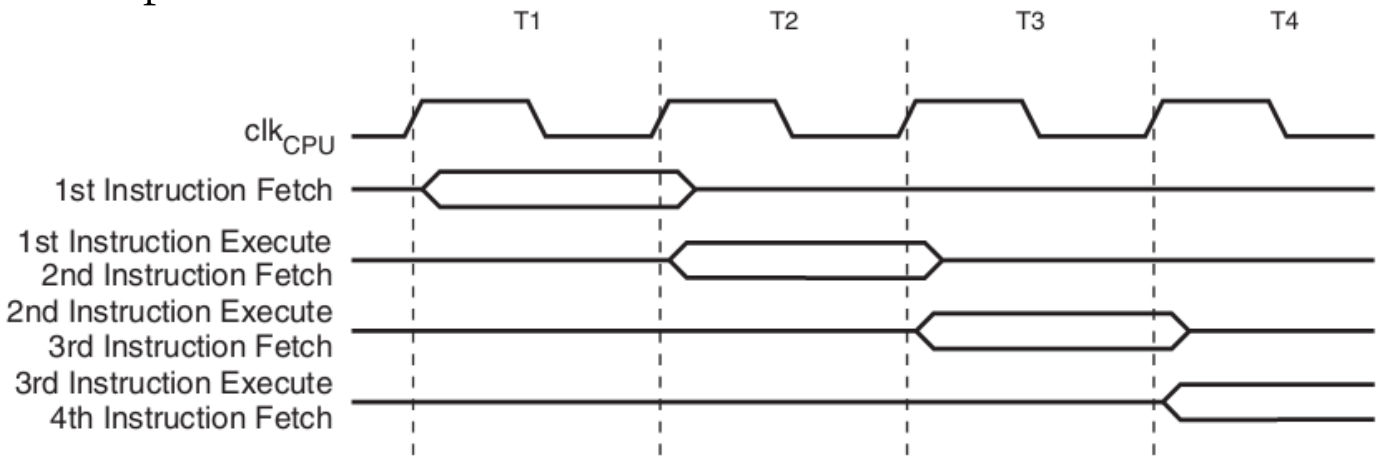

PF0 - PF7
PA0 - PA7
PC0 - PC7
XTAL1
XTAL2
RESET
VCC
GND
PORTF DRIVERS
PORTA DRIVERS
PORTC DRIVERS
DATA REGISTER PORTF
DATA DIR. REG. PORTF
DATA REGISTER PORTA
DATA DIR. REG. PORTA
DATA REGISTER PORTC
DATA DIR. REG. PORTC
8-BIT DATA BUS
AVCC
AGND
AREF
ADC
INTERNAL OSCILLATOR
CALIB. OSC
OSCILLATOR
OSCILLATOR
JTAG TAP
PROGRAM COUNTER
STACK POINTER
WATCHDOG TIMER
ON-CHIP DEBUG
PROGRAM FLASH
SRAM
MCU CONTROL REGISTER
TIMING AND CONTROL
BOUNDARY-SCAN
INSTRUCTION REGISTER
GENERAL PURPOSE REGISTERS
X
Y
Z
TIMER/ COUNTERS
PEN
PROGRAMMING LOGIC
INSTRUCTION DECODER
INTERRUPT UNIT
CONTROL LINES
ALU
EEPROM
STATUS REGISTER
USART0
SPI
USART1
TWO-WIRE SERIAL INTERFACE
ANALOG COMPARATOR
DATA REGISTER PORTE
DATA DIR. REG. PORTE
DATA REGISTER PORTB
DATA DIR. REG. PORTB
DATA REGISTER PORTD
DATA DIR. REG. PORTD
DATA REG. PORTG
DATA DIR. REG. PORTG
PORTE DRIVERS
PORTB DRIVERS
PORTD DRIVERS
PORTG DRIVERS
PE0 - PE7
PB0 - PB7
PD0 - PD7
PG0 - PG4

CPU Core


Data Bus 8-bit
Flash Program Memory
Program Counter
Status and Control
Instruction Register
Instruction Decoder
Control Lines
32 x 8 General Purpose Registrers
ALU
Direct Addressing
Indirect Addressing
Data SRAM
EEPROM
I/O Lines
Interrupt Unit
SPI Unit
Watchdog Timer
Analog Comparator
I/O Module1
I/O Module 2
I/O Module n

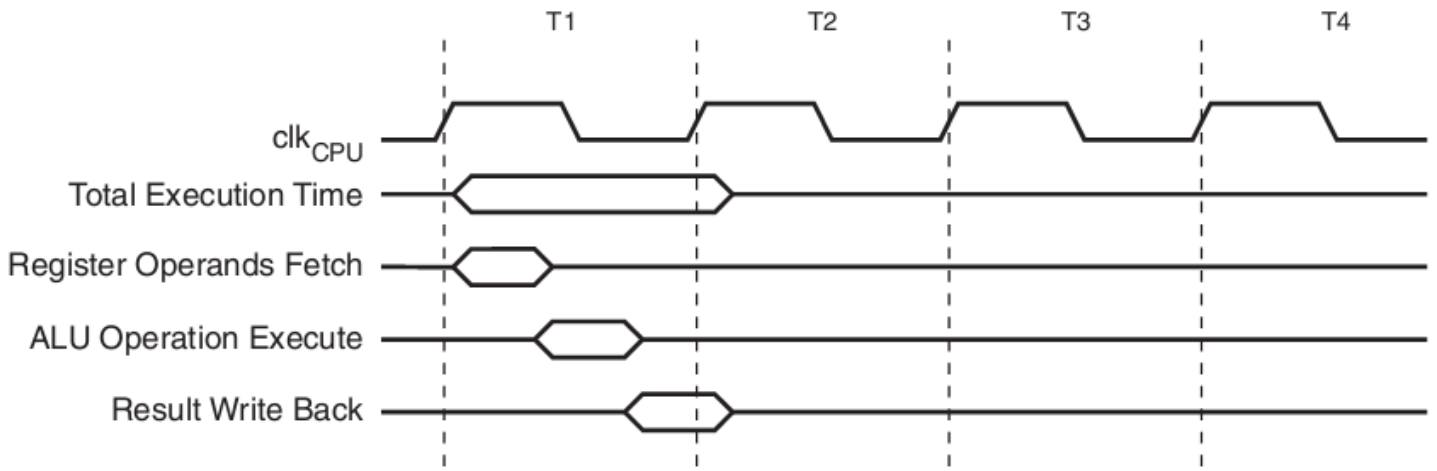
ALU I.

- Instruction Execution Timing
  - The AVR CPU is driven by the CPU clock, directly generated from the selected clock source for the chip. No internal clock division is used.
  - Parallel Instruction Fetches and Instruction Executions
  - 1 MIPS per MHz



ALU II.

- ALU Operation Timing
  - In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.



ALU III.

- The Status Register (SREG) contains information about the result of the most recently executed arithmetic instruction. It can be used for altering program flow. Status Register is updated after all ALU operations, as specified in the Instruction Set Reference.
  - **I – Global Interrupt Enable**: must be set for the interrupts to be enabled.
  - **T – Bit Copy Storage**: A bit from a register in the Register file can be copied into T, and a bit in T can be copied into a bit in a register in the Register. It can be used for conditional operations.
  - **H – Half Carry Flag**: indicates a half carry in some arithmetic operations. Half carry is useful in BCD arithmetic.
  - **S – Sign Bit**: is always an exclusive or between the negative flag N and the two's complement overflow flag V.
  - **V – Two's Complement Overflow Flag**: supports two's complement arithmetics.
  - **N – Negative Flag**: indicates a negative result in an arithmetic or logic operation. (MSB of the result)
  - **Z – Zero Flag**: indicates a zero result in an arithmetic or logic operation.
  - **C – Carry Flag**: indicates a carry in an arithmetic or logic operation.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

General Purpose Register File

- 32 general purpose working registers (8-bit)
- Each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers.
- Most of the instructions operating on the Register file have direct access to all registers, and most of them are single cycle instructions.
  - Only the LDI (Load Immediate) operation has got a restriction. It can only be used with the higher 16 registers (R16-R31).
  - The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the Data Space.

7 … 0	Addr.
R0	$00
R1	$01
R2	$02
…	
R13	$0D
R14	$0E
R15	$0F
R16	$10
R17	$11
…	
R26	$1A
R27	$1B
R28	$1C
R29	$1D
R30	$1E
R31	$1F

R26	$1A	X-register Low Byte
R27	$1B	X-register High Byte
R28	$1C	Y-register Low Byte
R29	$1D	Y-register High Byte
R30	$1E	Z-register Low Byte
R31	$1F	Z-register High Byte

Other important CPU registers

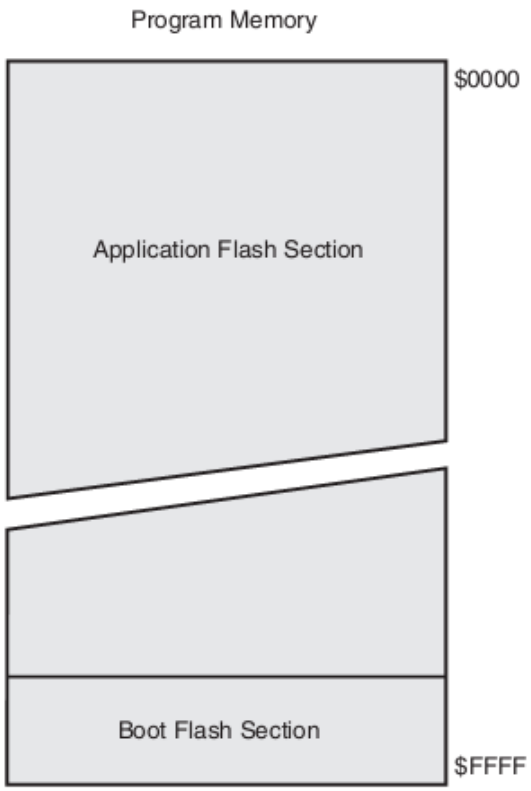
- **Program Counter** (PC): The ATmega128 Program Counter (PC) is 16 bits wide, thus addressing the 64K program memory locations.
- **Stack Pointer** (SP): The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.
- **MCU Control Register** (MCUCR): external memory enable, sleep mode enable, interrupt vector selection
- **MCU Control and Status Register** (MCUCSR): JTAG Interface status and control
- **Oscillator Calibration Value** (OSCCAL): internal RC oscillator calibration value
- **XTAL Divide Control Register** (XDIV): source clock frequency divider
- **External Memory Control Register A & B** (XMCRA, B): external memory control

Fuse bits

- Fuse bits are the control bits of program memory. They can be programmed by the programmer device.
  - ATmega103 compatibility mode
  - Watchdog Timer always on
  - Enable Debug, JTAG, SPI programming modes
  - Clock source and start-up time settings
  - Enable Brown out detector
  - Reset vector, boot memory settings
  - EEPROM content saving from chip erase

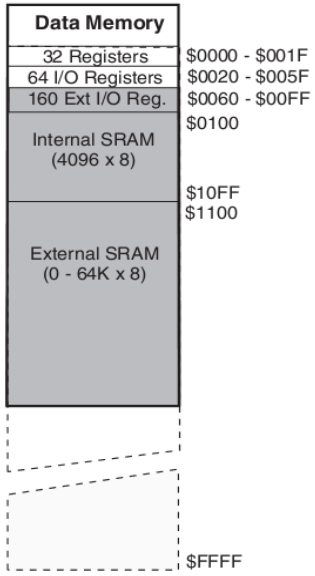
Program memory

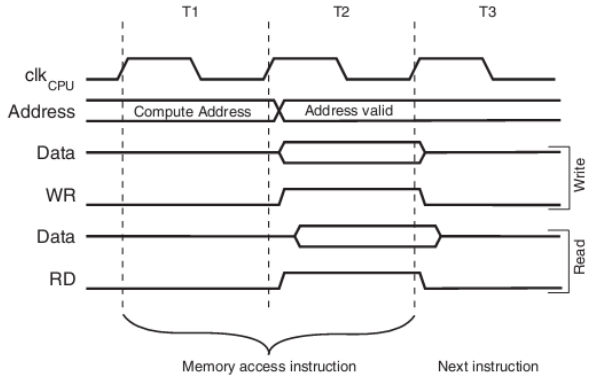
- 128 kB On-chip In-System Reprogrammable Flash
- Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 64Kx16 (16-bit program counter is enough)
- 10 000 write/erase cycles
- For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.
- Programming modes
  - SPI, JTAG, Parallel Programming
  - Self-Programming



Data Memory (SRAM)

- The first 4352 Data Memory locations address both the Register file, the I/O Memory, Extended I/O Memory, and the internal data SRAM.
- The first 32 locations address the Register file
  - It's just for addressing. The registers are implemented physically in the CPU core.
- The next 64 locations are the standard I/O memory
  - Directly addressed by the I/O operations (IN and OUT, 6-bit address space)
  - The first 32 I/O registers (5 bit address space) can be accessed by bit addressing
- Then 160 locations of Extended I/O memory
  - The ATmega128 is a complex microcontroller with more peripheral units than can be supported within the 64 location
  - For this memory space, only the general data transfer operations can be used (ST/STS/STD and LD/LDS/LDD), not the I/O operations
- The next 4096 locations address the internal data SRAM
- This optional external SRAM will occupy an area in the remaining address locations in the 64K address space.
  - The external SRAM is accessed using the same instructions as for the internal data memory access.
  - The usage of the external SRAM has to be enabled in the MCUCR.
  - Accessing external SRAM takes one additional clock cycle per byte compared to access of the internal SRAM.





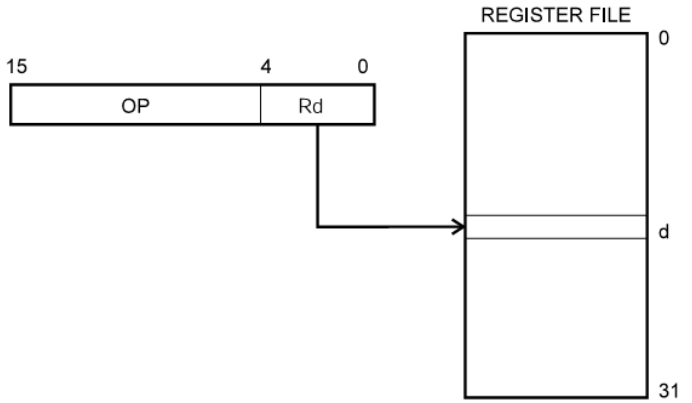
Memory Addressing Modes I.

- Data Memory Addressing Modes
  - Immediate Addressing Data
  - Bit Addressing
  - Register Direct, Single Register
  - Register Direct, Two Registers
  - I/O Direct Addressing
  - Direct Data Addressing
  - Data Indirect Addressing
  - Data Indirect with Displacement
  - Data Indirect Addressing with Pre-Decrement
  - Data Indirect Addressing with Post-Increment
- Program Memory Addressing Modes
  - Program Memory Constant Addressing
  - Program Memory Constant Addressing with Post-Increment
  - Direct Program Memory Addressing
  - Indirect Program Memory Addressing
  - Relative Program Memory Addressing

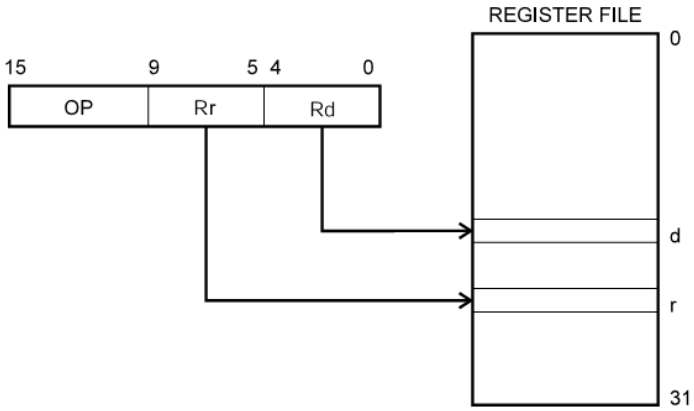
Memory Addressing Modes II.

- Immediate Addressing Data
  - Operands are a register and a constant values (e.g., LDI R16, 0x55)
- Bit Addressing
  - The operand is a bit of a register (e.g., BCLR 7)
  - Not all registers can be bit addressed.
- Direct Single Register Addressing
  - The operand is contained in register d (Rd). (e.g.,: INC R16)
- Direct Register Addressing, Two Registers
  - Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd). (e.g.,: ADD R17,R16)

Direct Single Register Addressing



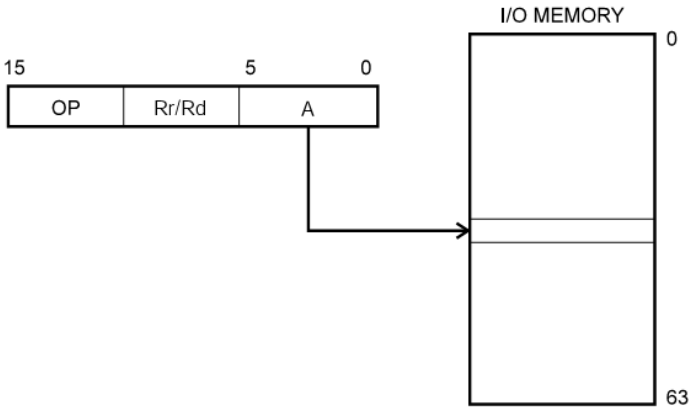
Direct Register Addressing, Two Registers



Memory Addressing Modes III.

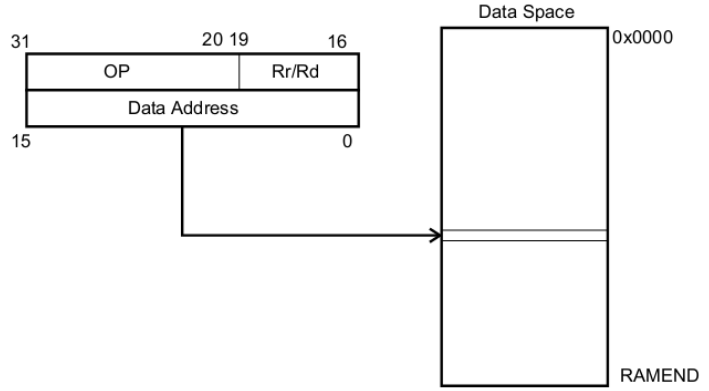
- I/O Direct Addressing
  - Operand address is contained in 6 bits of the instruction word. R_r/R_d is the destination or source register address. (e.g.,: IN R16,PINE)

I/O Direct Addressing



- Direct Data Addressing
  - A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register. (e.g., LDS R16, 0x0500)

Direct Data Addressing

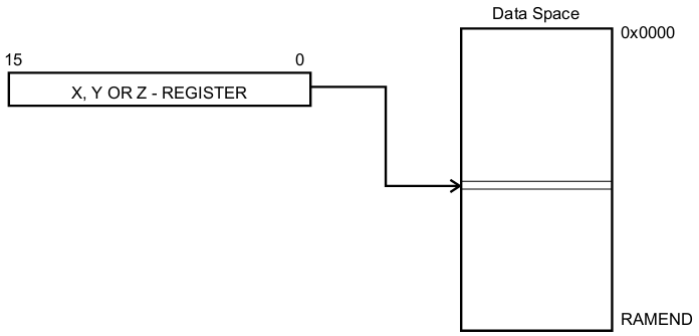


Memory Addressing Modes IV.

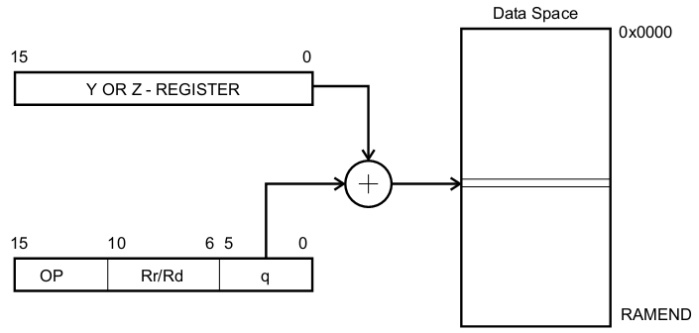
- Data Indirect Addressing
  - Operand addresses are the contents of the X-, Y-, or the Z-register and R_r/R_d registers. (e.g., LD R16, X)

- Data Indirect with Displacement
  - Operand address is the result of the Y- or Z-register contents added to the address contained in 6 bits of the instruction word. Rd/Rr specify the destination or source register. (e.g., LDD R16, Y+0x10)

Data Indirect Addressing



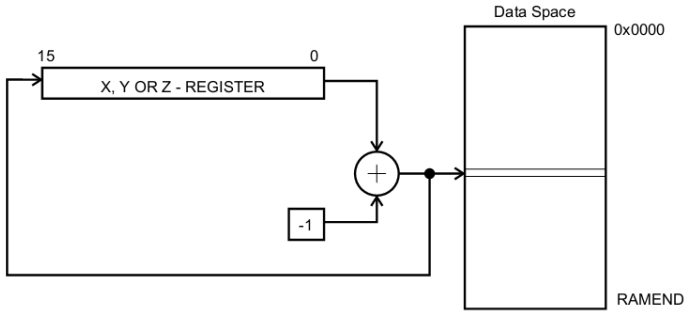
Data Indirect with Displacement



Memory Addressing Modes V.

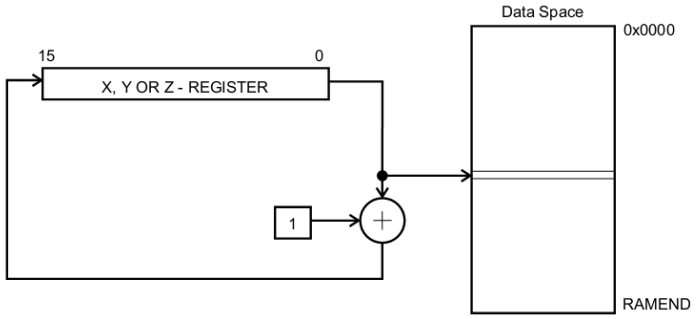
- Data Indirect Addressing with Pre-decrement
  - The X,- Y-, or the Z-register is decremented before the operation. Operand address is the decremented contents of the X-, Y-, or the Z-register.(e.g., LD R16, -X)

Data Indirect Addressing with Pre-decrement



- Data Indirect Addressing with Post-increment
  - The X-, Y-, or the Z-register is incremented after the operation. Operand address is the content of the X-, Y-, or the Z-register prior to incrementing. (e.g., LD R16, +X)

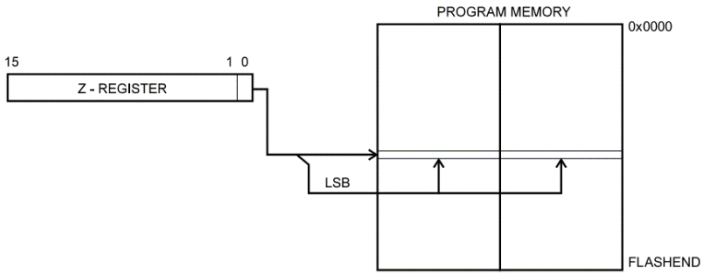
Data Indirect Addressing with Post-increment



Memory Addressing Modes VI.

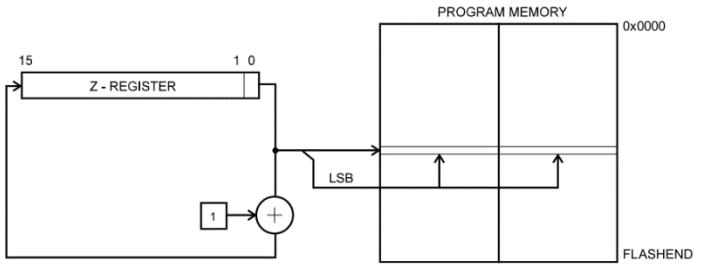
- Program Memory Constant Addressing
  - Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. For LPM, the LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1). (e.g., LPM R16,Z)

Program Memory Constant Addressing



- Program Memory Addressing with Post-increment
  - Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. The LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1).

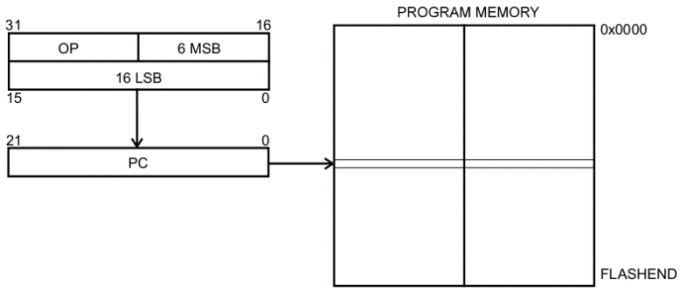
Program Memory Addressing with Post-increment



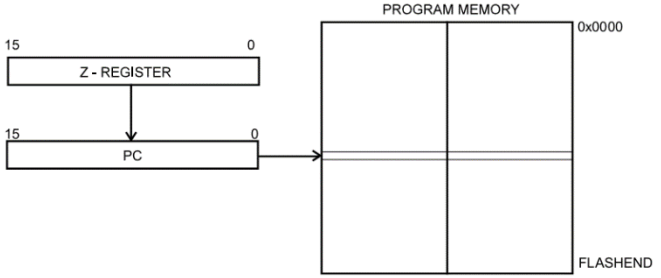
Memory Addressing Modes VII.

- Direct Program Memory Addressing
  - Program execution continues at the address immediate in the instruction word. (e.g., JMP here)
- Indirect Program Memory Addressing
  - Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register).
- Relative Program Memory Addressing
  - Program execution continues at address PC + k + 1. The relative address k is from -2048 to 2047. (e.g., RJMP 0x010)

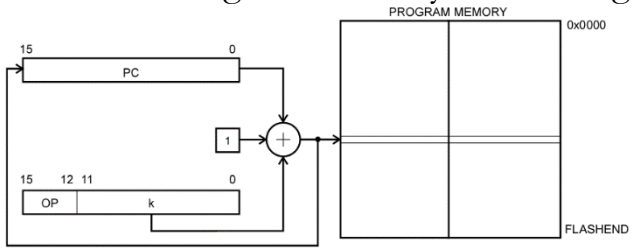
Direct Program Memory Addressing



Indirect Program Memory Addressing



Relative Program Memory Addressing



Instruction Set Summary

- Arithmetic and Logic Instructions
  - Addition, subtraction, multiplication
  - AND, OR, XOR, two's complement
  - Increment, decrement, set, clear
- Branch Instructions
  - Unconditional jumps
  - Conditional jumps according to the SREG bits
  - Subroutine calls and returns
- Data Transfer Instructions
  - Between data memory and registers
  - Between program memory and registers
  - Between I/O and registers
- Bit and bit-test instructions
  - Shift, rotate swap nibbles
  - SREG, I/O, Rr register bits clear and set
- MCU control
  - No-operation, sleep mode, watchdog reset
  - Break (only for debugging)

The End

Thank you for your attention!