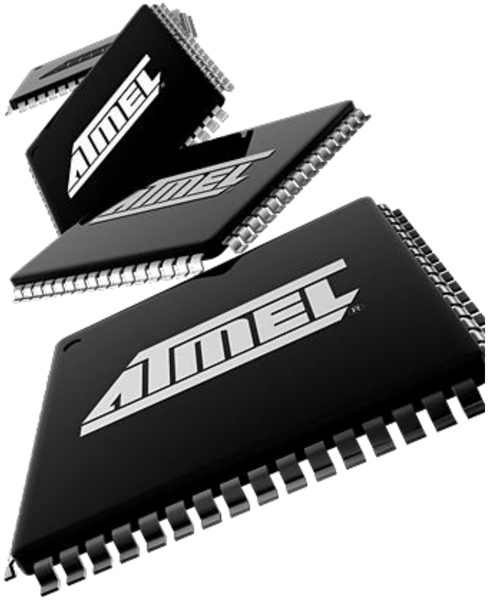
Principles and Applications of Microcontrollers

Yan-Fu Kuo
Dept. of Biomechatronics Engineering
National Taiwan University

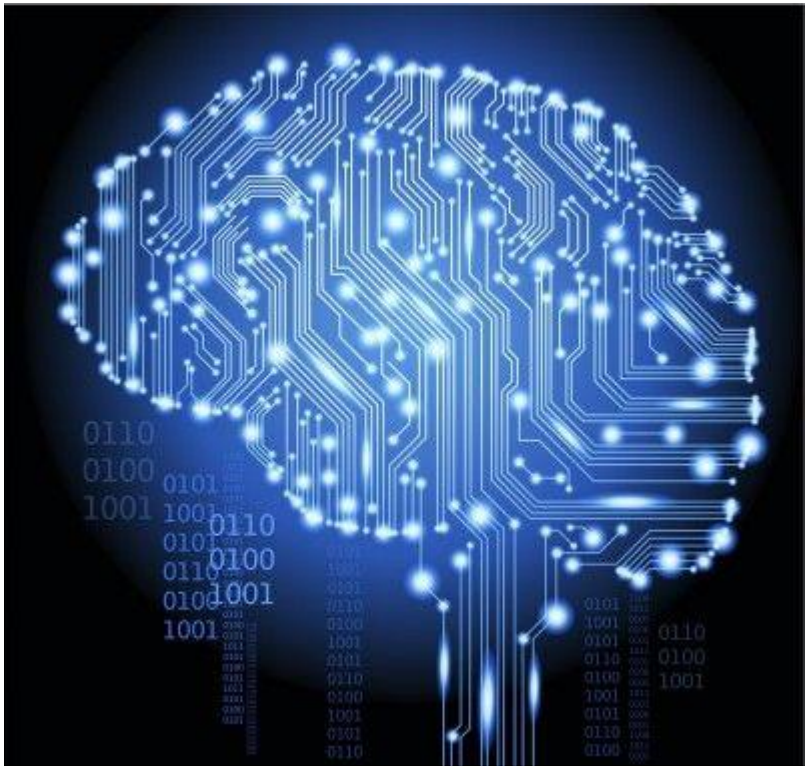
Today:

- AVR assembly language
- Machine code
- Status register



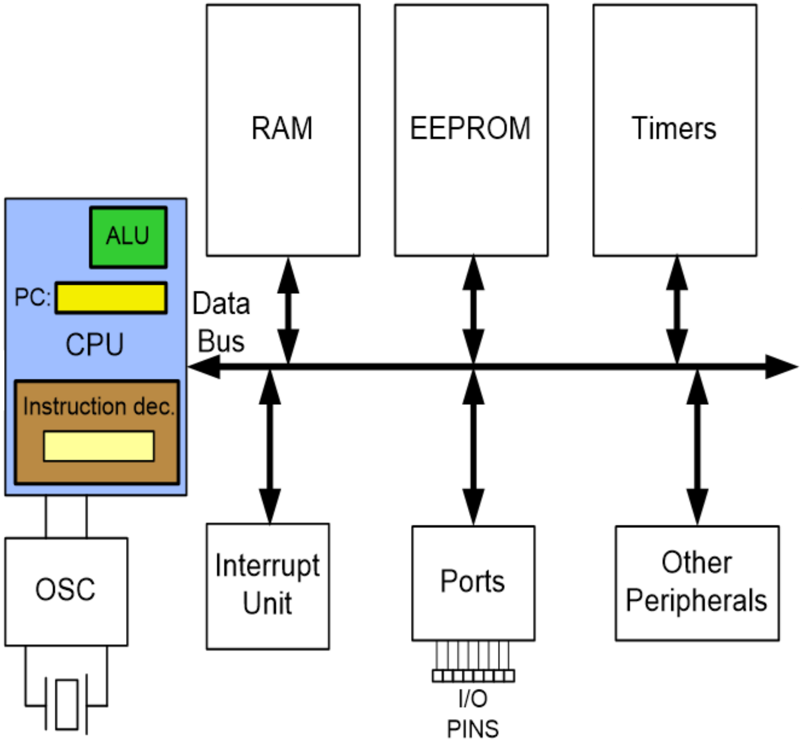
Outline

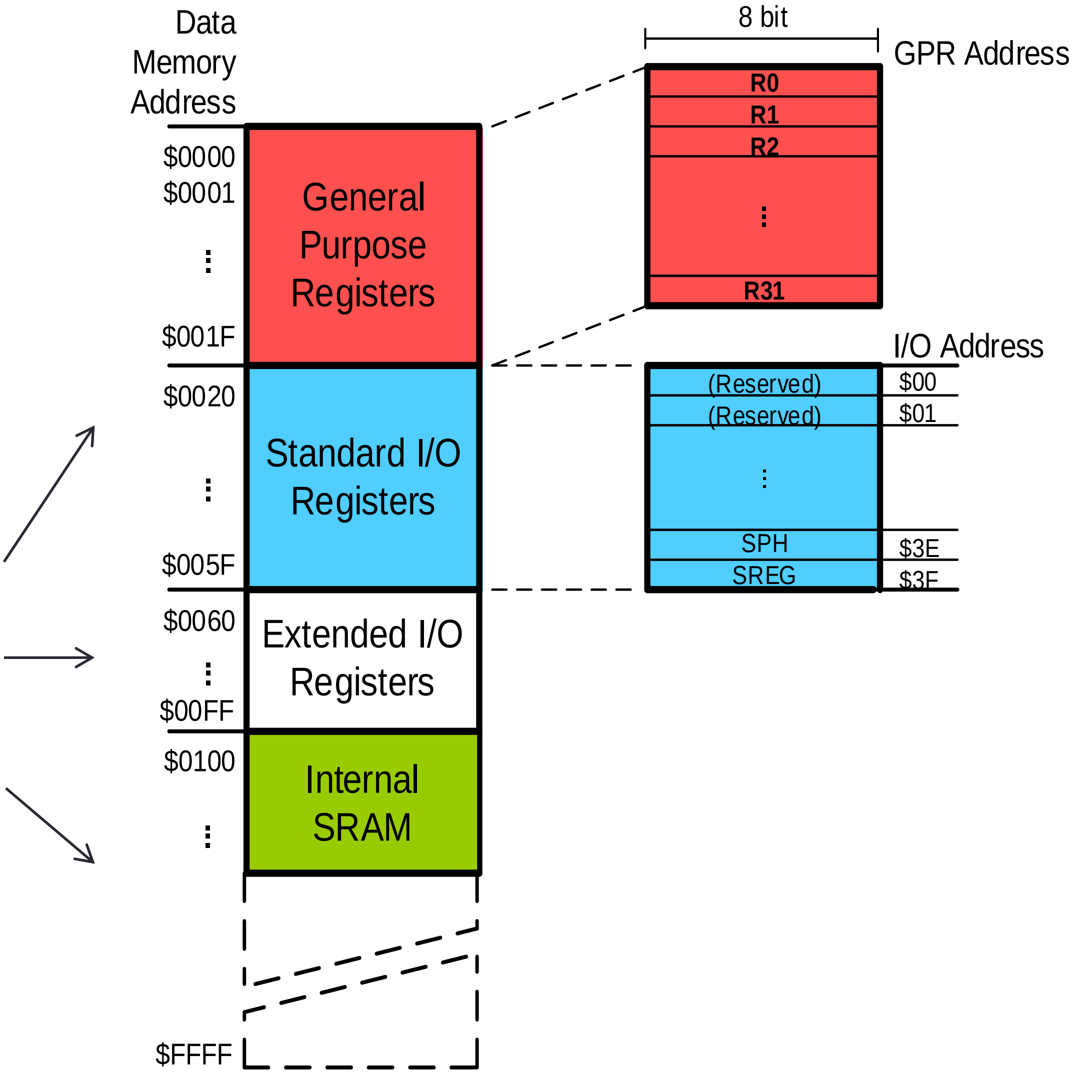
- Accessing data memory
- AVR assembly language
  - Structure of AVR assembly
  - Machine code
- Status register
- Getting started



Data Memory Access

- How to access data memory outside the I/O?





Accessing Data Memory – LDS & STS

- LDS – load data from space
- Syntax: **LDS Rd, k**

  where 0 ≤ d ≤ 31, k between $0000 to $FFFF
- Example:
  - **LDS R1, 0x60 ;R1= [0x60]**

- STS – sto
- Syntax: S

  where 0 ≤
- Example:
  - **STS 0x60, R15 ;[0x60] = R15**

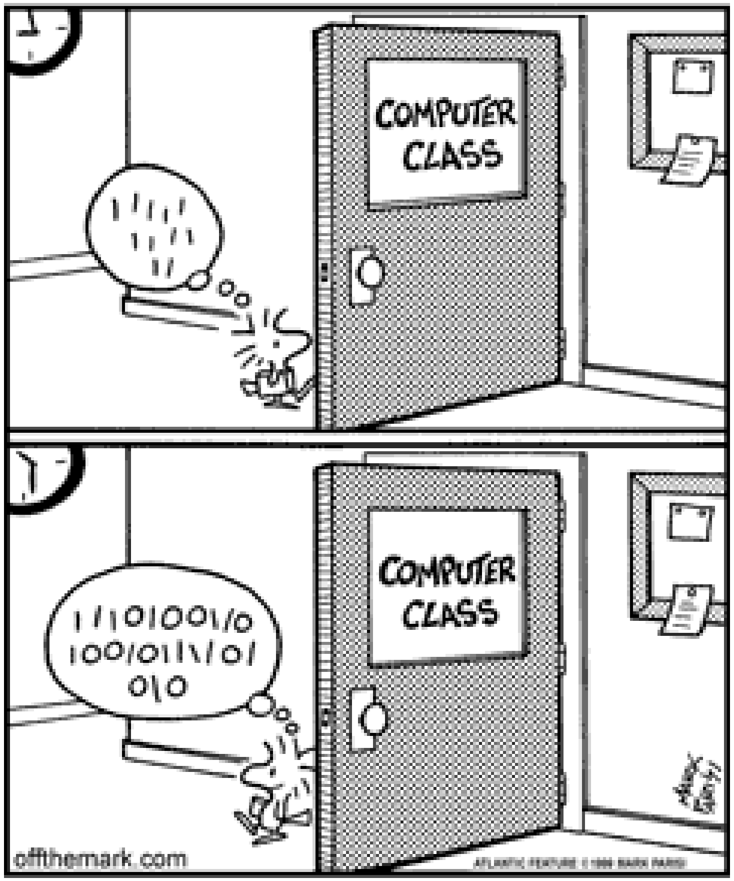
Example: Add contents of location 0x90 to contents of location 0x95 and store the result in location 0x313.

Solution:

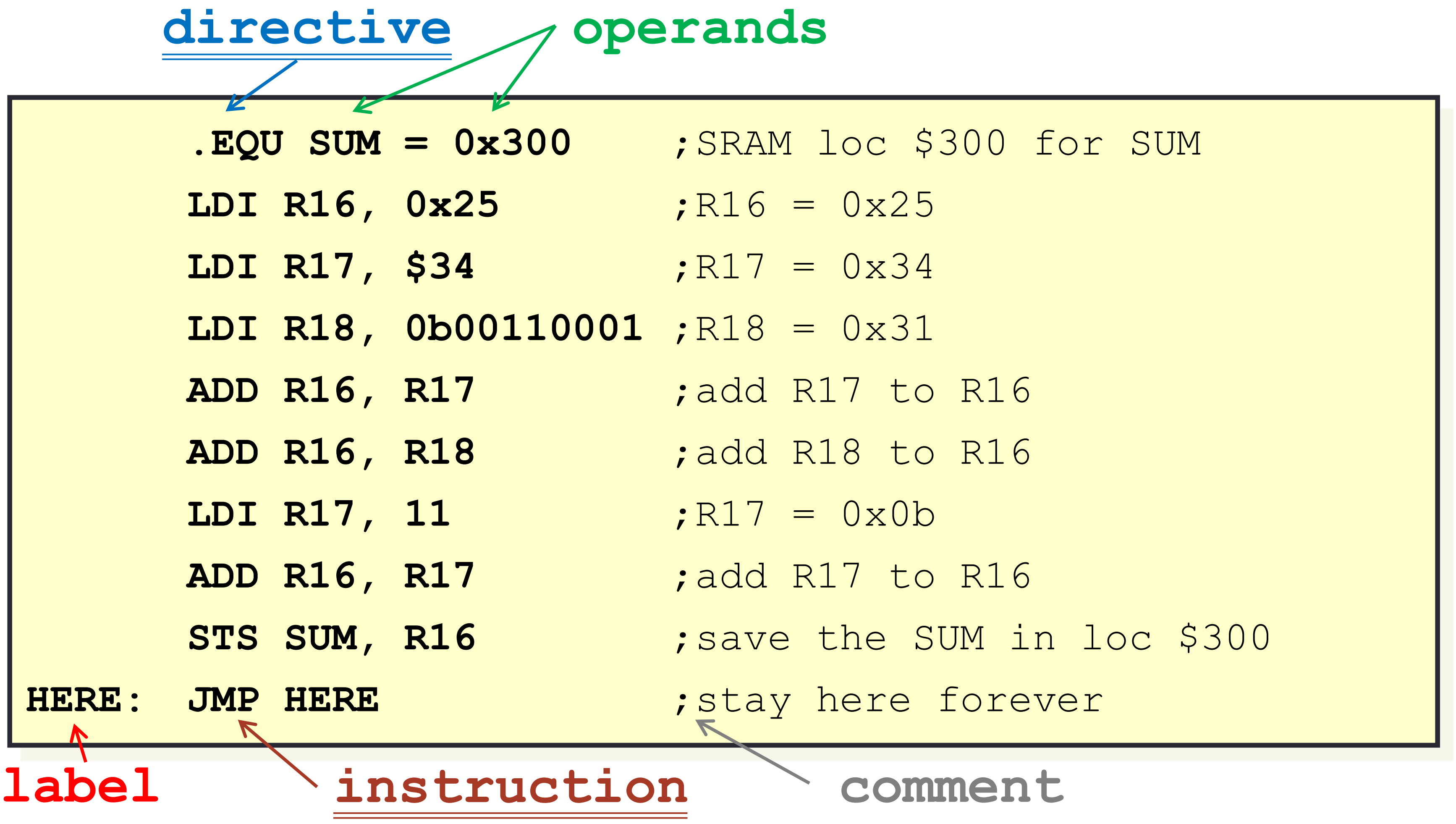
```
LDS   R20, 0x90        ;R20 = [0x90]
LDS   R21, 0x95        ;R21 = [0x95]
ADD   R20, R21         ;R20 = R20 + R21
STS   0x313, R20       ;[0x313] = R20
```

Outline (Cont'd)

- Accessing data memory
- AVR assembly language
  - Structure of AVR assembly
  - Machine code
- Status register
- Getting started



An Example Assembly Program



```
         .EQU SUM = 0x300     ;SRAM loc $300 for SUM
         LDI R16, 0x25        ;R16 = 0x25
         LDI R17, $34         ;R17 = 0x34
         LDI R18, 0b00110001  ;R18 = 0x31
         ADD R16, R17         ;add R17 to R16
         ADD R16, R18         ;add R18 to R16
         LDI R17, 11          ;R17 = 0x0b
         ADD R16, R17         ;add R17 to R16
         STS SUM, R16         ;save the SUM in loc $300
HERE:    JMP HERE             ;stay here forever
```

Assembler Directives – .EQU & .SET

- Syntax: `.EQU` *name* = *value*
- Example:

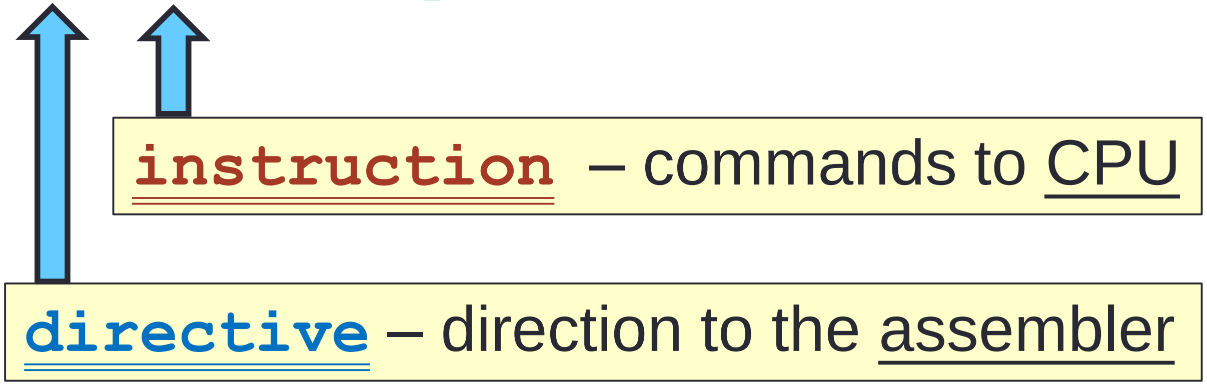
```
.EQU    COUNT = 0x25
LDI     R21, COUNT              ;R21 = 0x25
LDI     R22, COUNT + 3          ;R22 = 0x28
```

- Syntax: `.SET` *name* = *value*
- Example:

```
.SET    COUNT = 0x25
LDI     R21, COUNT              ;R21 = 0x25
LDI     R22, COUNT + 3          ;R22 = 0x28
.SET    COUNT = 0x19
LDI     R21, COUNT              ;R21 = 0x19
```

Structure of Assembly Language

`[label:] MNEMONIC [operands][;comments]`



instruction – commands to CPU

directive – direction to the assembler

- directive does NOT generate any machine code
- operands are data items being manipulated
- For assembly, is it case sensitive?

Frequently Used Instructions

Arithmetic and logic

a+b	ADD
a-b	SUB
a&b	AND
a|b	OR
a++	INC
a--	DEC
-a	NEG
a=0	CLR

Move

reg1=reg2	MOV
reg=17	LDI
reg=mem	LDS
mem=reg	STS
periperal	IN
peripheral	OUT
heap	PUSH
heap	POP

Bit operation and others

a<<1	LSL
a>>1	LSR,
Ø C (not avail. In C)	ROL, ROR
Status bits	SEI, CLI, CLZ…
No op.	NOP

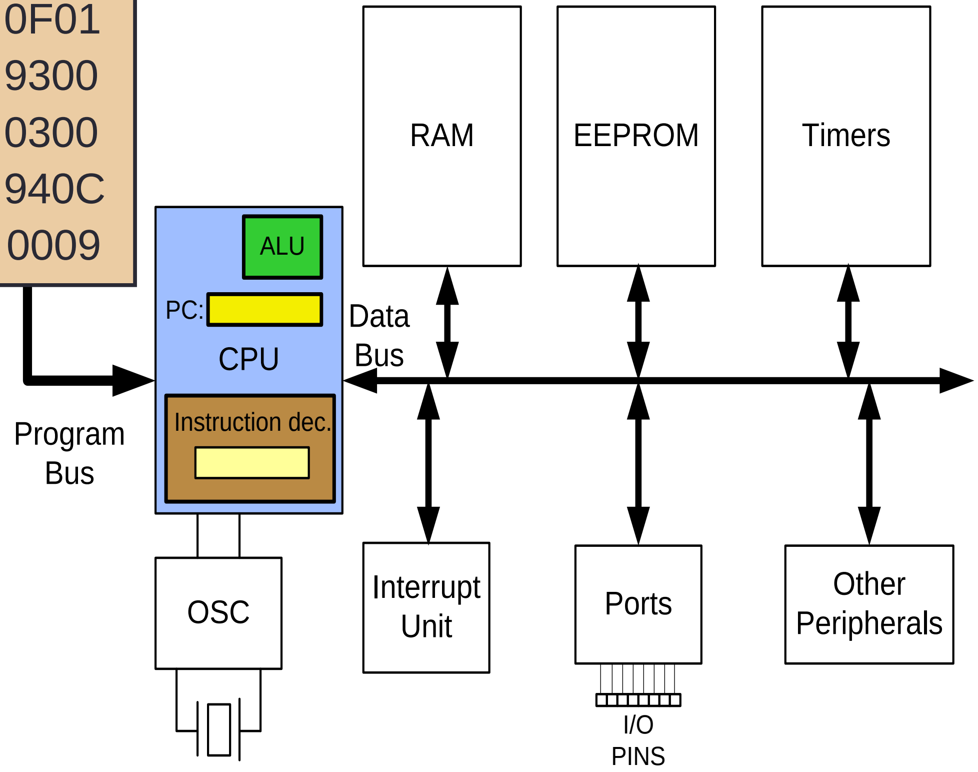
From: AVR 8-bit instruction set

Machine Code

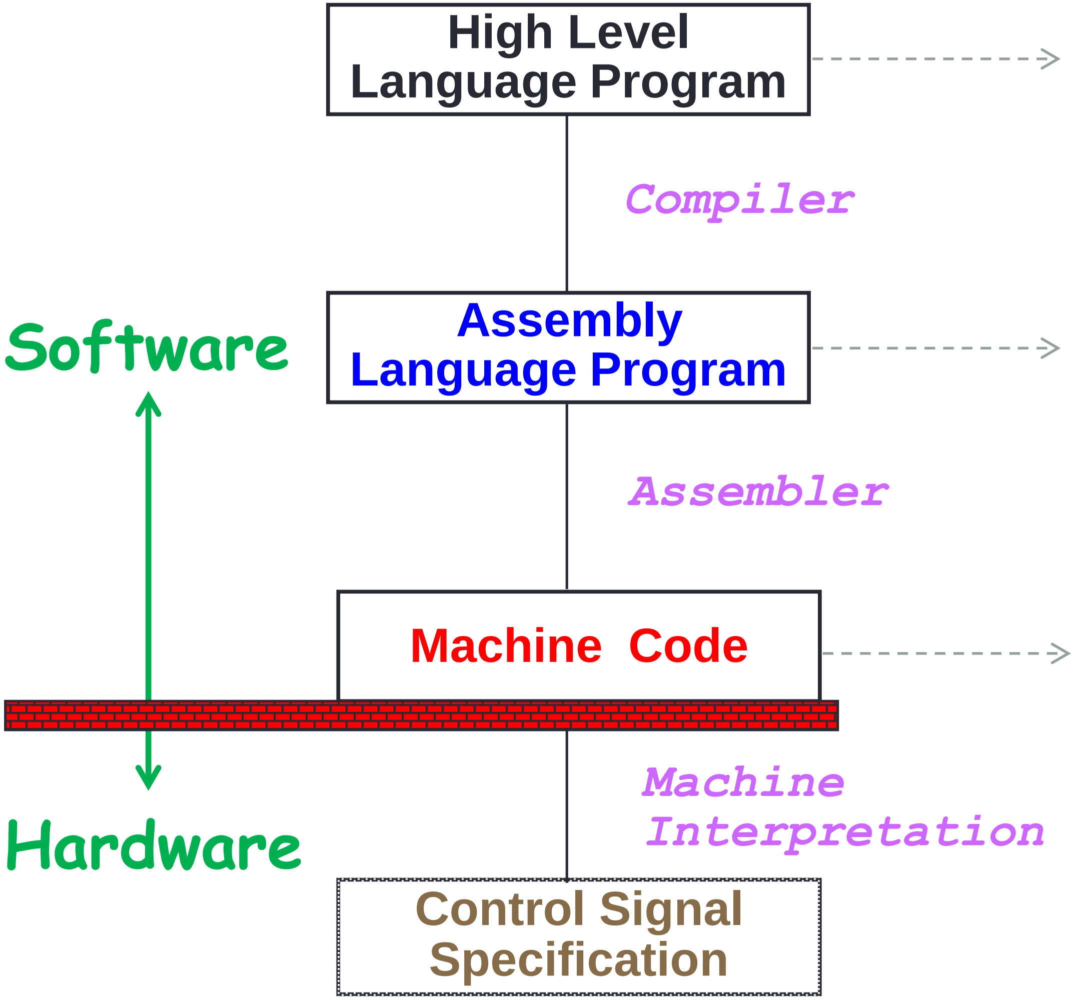
- Assembly code is turned into machine code
- Size of each machine code: 16 or 32 bits

```
      LDI  R16, 0x25
      LDI  R17, $34
      LDI  R18, 0x31
      ADD  R16, R17
      ADD  R16, R18
      LDI  R17, 11
      ADD  R16, R17
      STS  0x300, R16
HERE:JMP HERE
```

00	E205
01	E314
02	E321
03	0F01
04	0F02
05	E01B
06	0F01
07	9300
08	0300
09	940C
0A	0009



Levels of Computer Language



High Level Language Program

Compiler

Assembly Language Program

Assembler

Machine Code

Machine Interpretation

Control Signal Specification

Software

Hardware

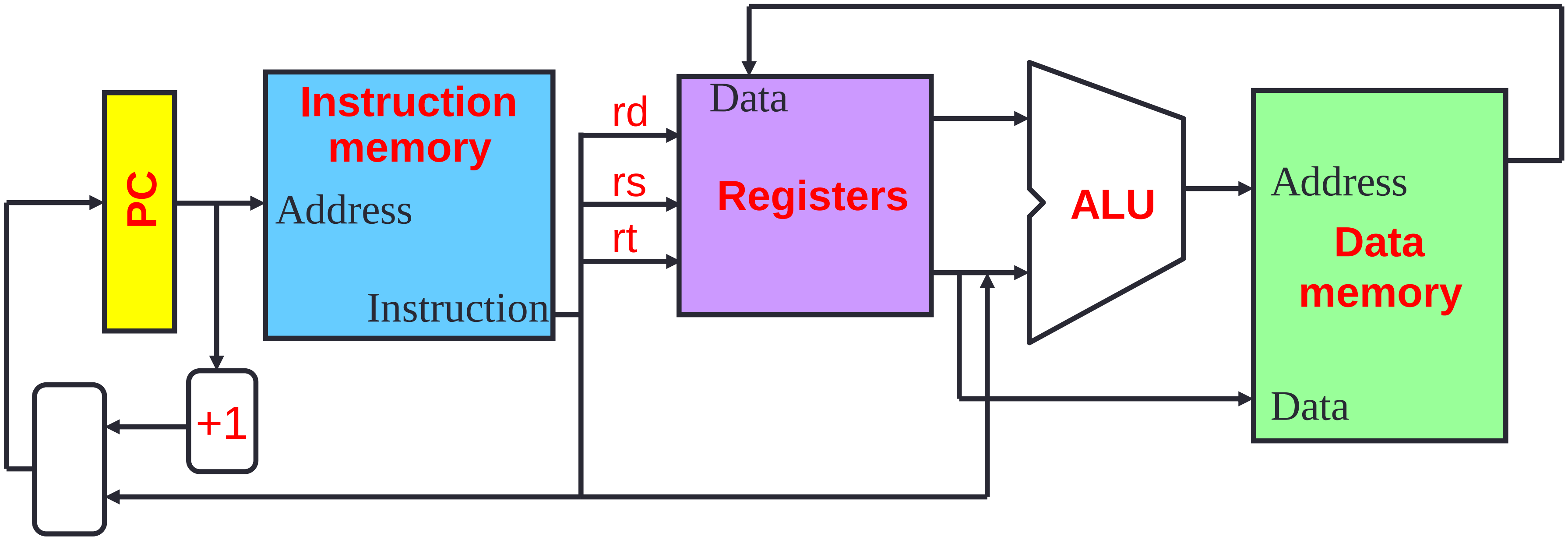
```
A = B + C;
D = E + F;
```

```
LDI  R16, 0x25
LDI  R17, $34
LDI  R18, 0x31
```

```
0000 1001 1100 0110 1010
1010 1111 0101 1000 0000
1100 0110 1010 1111 0101
```

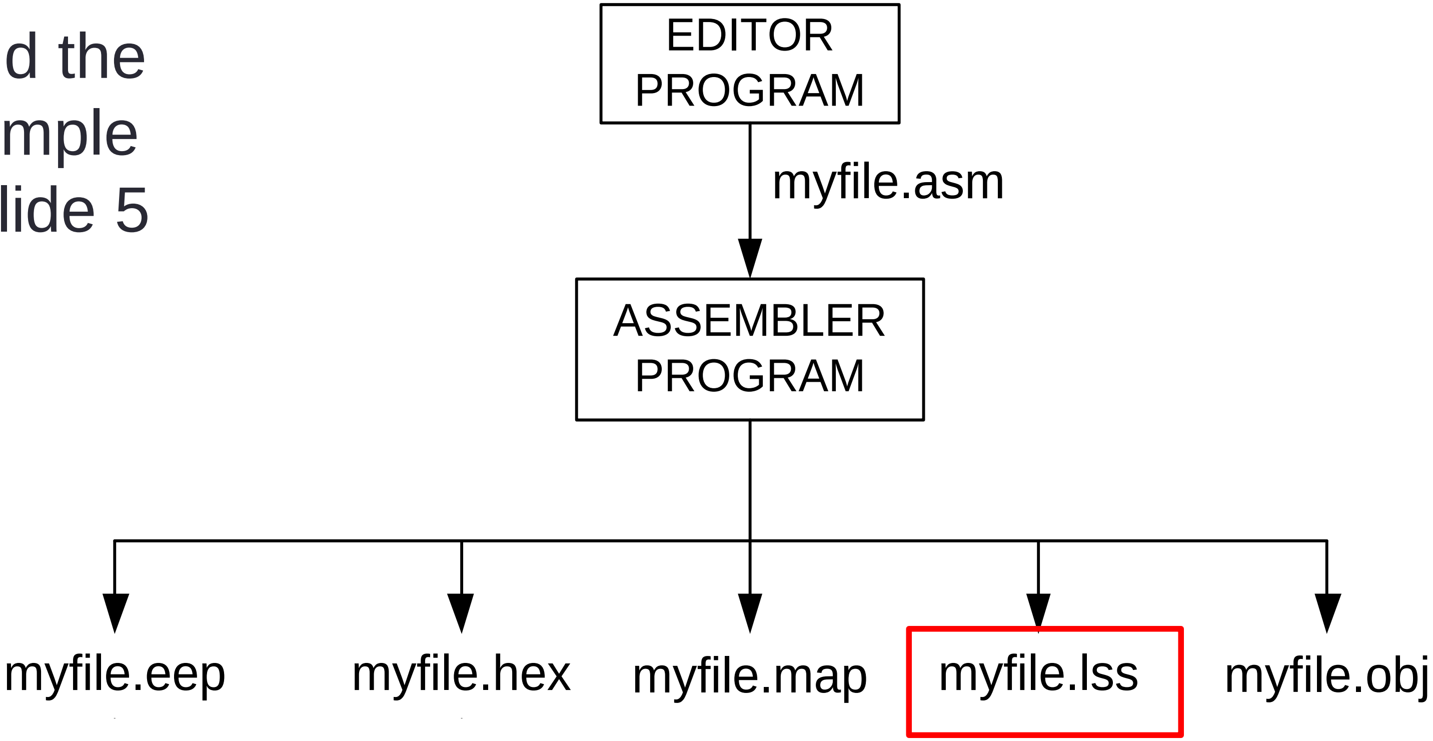
Instruction Set Architecture

- The instructions are actually realized in hardware
- Wires are connected between each components

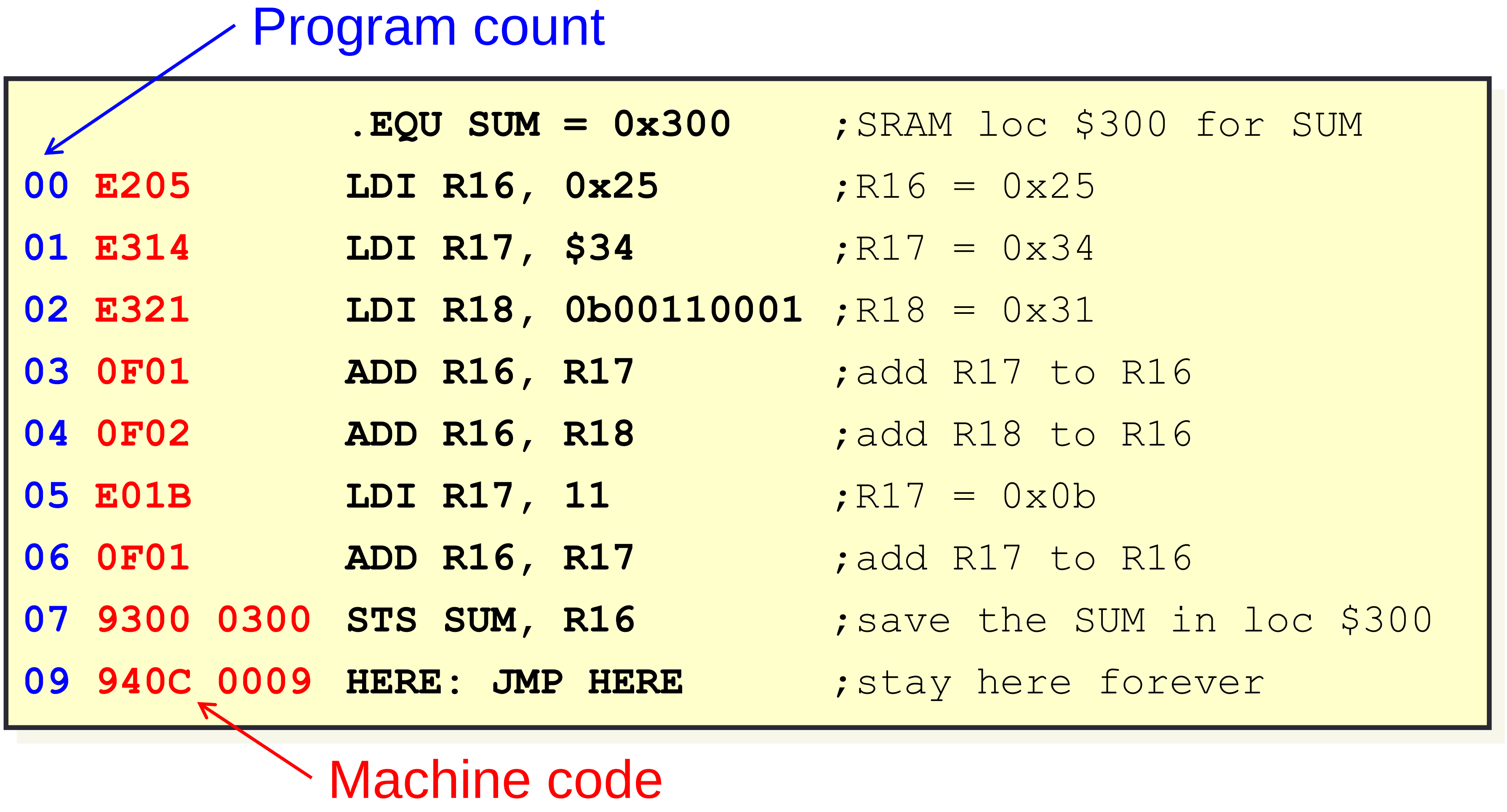


Assembler

- Build the example in slide 5



Example List (.lss) File



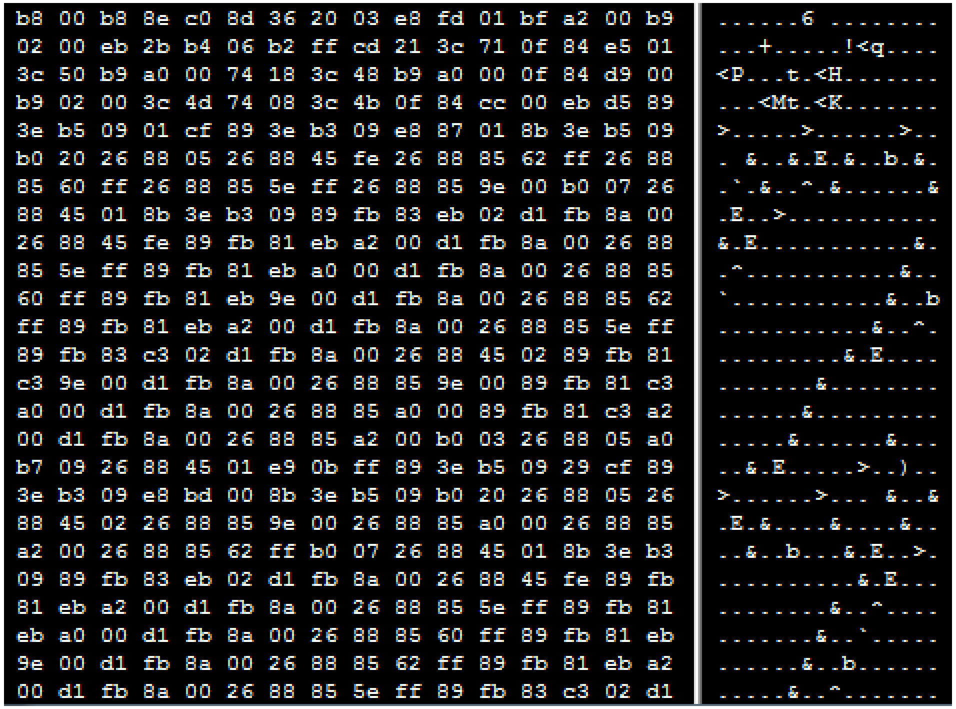
Program count

```
                .EQU SUM = 0x300      ;SRAM loc $300 for SUM
00 E205         LDI R16, 0x25         ;R16 = 0x25
01 E314         LDI R17, $34          ;R17 = 0x34
02 E321         LDI R18, 0b00110001 ;R18 = 0x31
03 0F01         ADD R16, R17          ;add R17 to R16
04 0F02         ADD R16, R18          ;add R18 to R16
05 E01B         LDI R17, 11           ;R17 = 0x0b
06 0F01         ADD R16, R17          ;add R17 to R16
07 9300 0300 STS SUM, R16          ;save the SUM in loc $300
09 940C 0009 HERE: JMP HERE        ;stay here forever
```

Machine code

Outline (Cont’d)

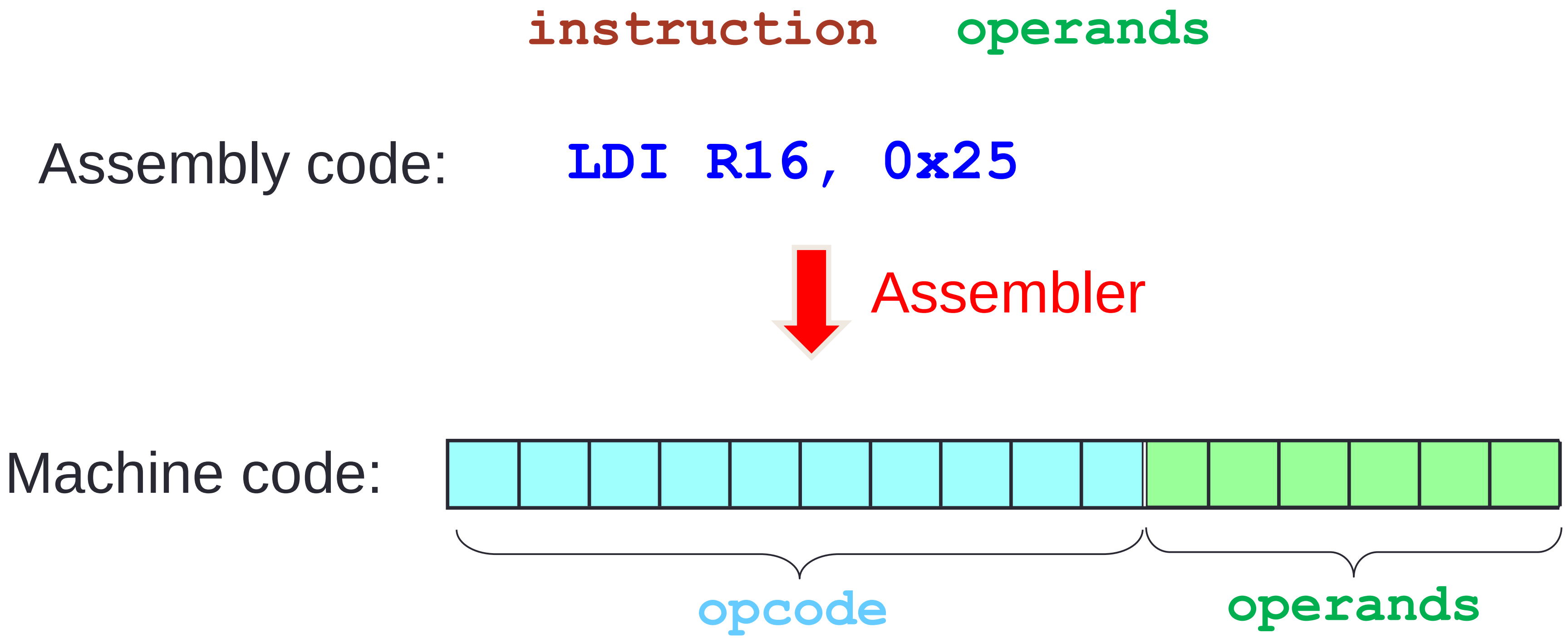
- Accessing data memory
- AVR assembly language
  - Structure of AVR assembly
  - Machine code
- Status register
- Getting started



```
b8 00 b8 8e c0 8d 36 20 03 e8 fd 01 bf a2 00 b9   ......6 ........
02 00 eb 2b b4 06 b2 ff cd 21 3c 71 0f 84 e5 01   ...+.....!<q....
3c 50 b9 a0 00 74 18 3c 48 b9 a0 00 0f 84 d9 00   <P...t.<H.......
b9 02 00 3c 4d 74 08 3c 4b 0f 84 cc 00 eb d5 89   ...<Mt.<K.......
3e b5 09 01 cf 89 3e b3 09 e8 87 01 8b 3e b5 09   >.....>......>..
b0 20 26 88 05 26 88 45 fe 26 88 85 62 ff 26 88   . &..&.E.&..b.&.
85 60 ff 26 88 85 5e ff 26 88 85 9e 00 b0 07 26   .`.&..^.&......&
88 45 01 8b 3e b3 09 89 fb 83 eb 02 d1 fb 8a 00   .E..>...........
26 88 45 fe 89 fb 81 eb a2 00 d1 fb 8a 00 26 88   &.E...........&.
85 5e ff 89 fb 81 eb a0 00 d1 fb 8a 00 26 88 85   .^...........&..
60 ff 89 fb 81 eb 9e 00 d1 fb 8a 00 26 88 85 62   `...........&..b
ff 89 fb 81 eb a2 00 d1 fb 8a 00 26 88 85 5e ff   ...........&..^.
89 fb 83 c3 02 d1 fb 8a 00 26 88 45 02 89 fb 81   .........&.E....
c3 9e 00 d1 fb 8a 00 26 88 85 9e 00 89 fb 81 c3   .......&........
a0 00 d1 fb 8a 00 26 88 85 a0 00 89 fb 81 c3 a2   ......&.........
00 d1 fb 8a 00 26 88 85 a2 00 b0 03 26 88 05 a0   .....&......&...
b7 09 26 88 45 01 e9 0b ff 89 3e b5 09 29 cf 89   ..&.E.....>..)..
3e b3 09 e8 bd 00 8b 3e b5 09 b0 20 26 88 05 26   >......>... &..&
88 45 02 26 88 85 9e 00 26 88 85 a0 00 26 88 85   .E.&....&....&..
a2 00 26 88 85 62 ff b0 07 26 88 45 01 8b 3e b3   ..&..b...&.E..>.
09 89 fb 83 eb 02 d1 fb 8a 00 26 88 45 fe 89 fb   ..........&.E...
81 eb a2 00 d1 fb 8a 00 26 88 85 5e ff 89 fb 81   ........&..^....
eb a0 00 d1 fb 8a 00 26 88 85 60 ff 89 fb 81 eb   .......&..`.....
9e 00 d1 fb 8a 00 26 88 85 62 ff 89 fb 81 eb a2   ......&..b......
00 d1 fb 8a 00 26 88 85 5e ff 89 fb 83 c3 02 d1   .....&..^.......
```

Composition of Machine Codes

- The assembler generates 16-bit or 32-bit machine codes
- Each machine code consists of 2 fields: opcode and operands



instruction operands

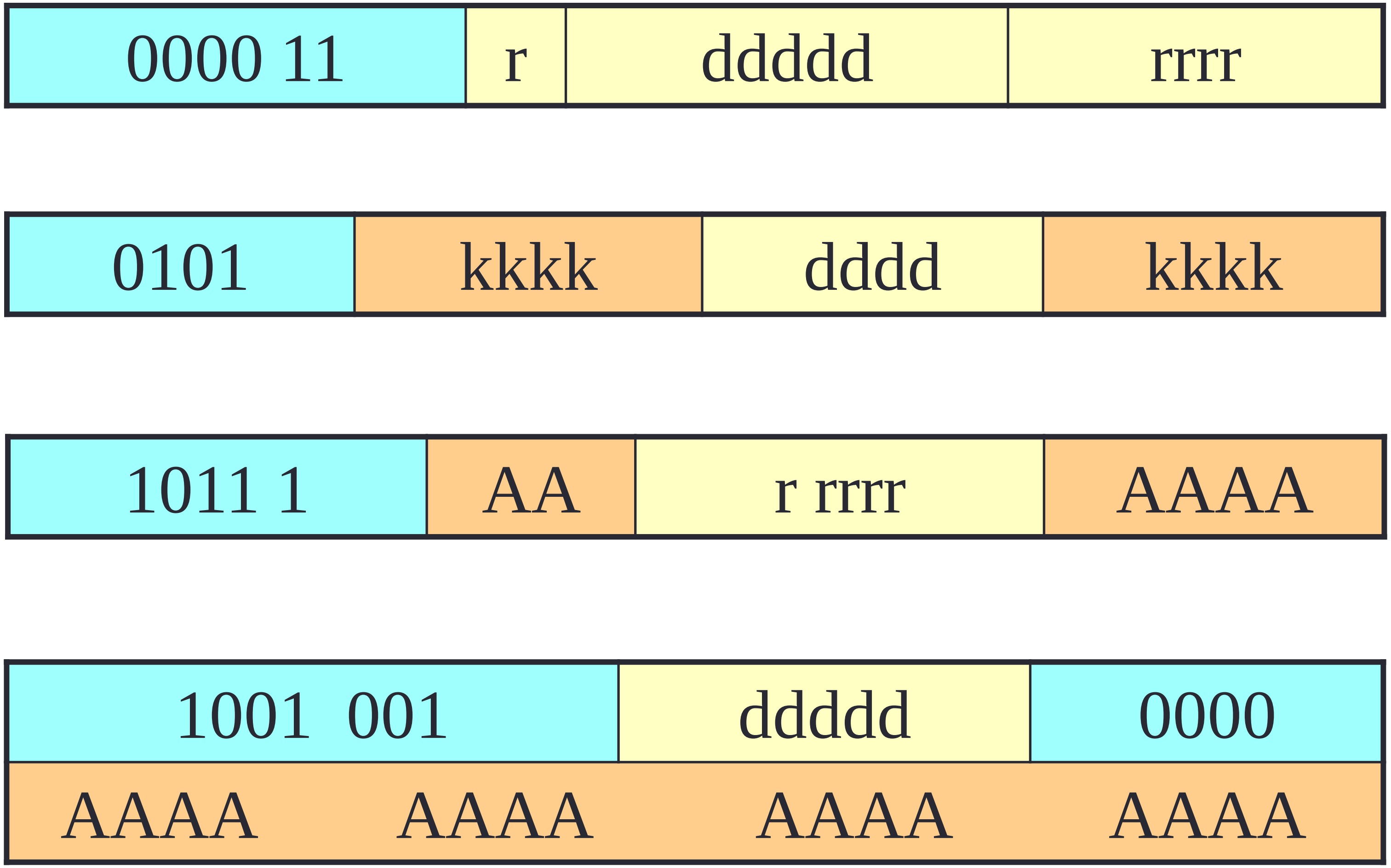
Assembly code: `LDI R16, 0x25`

Assembler

Machine code:

opcode operands

Some AVR Machine Code Examples



ADD: 0000 11 | r | ddddd | rrrr

SUBI: 0101 | kkkk | dddd | kkkk

OUT: 1011 1 | AA | r rrrr | AAAA

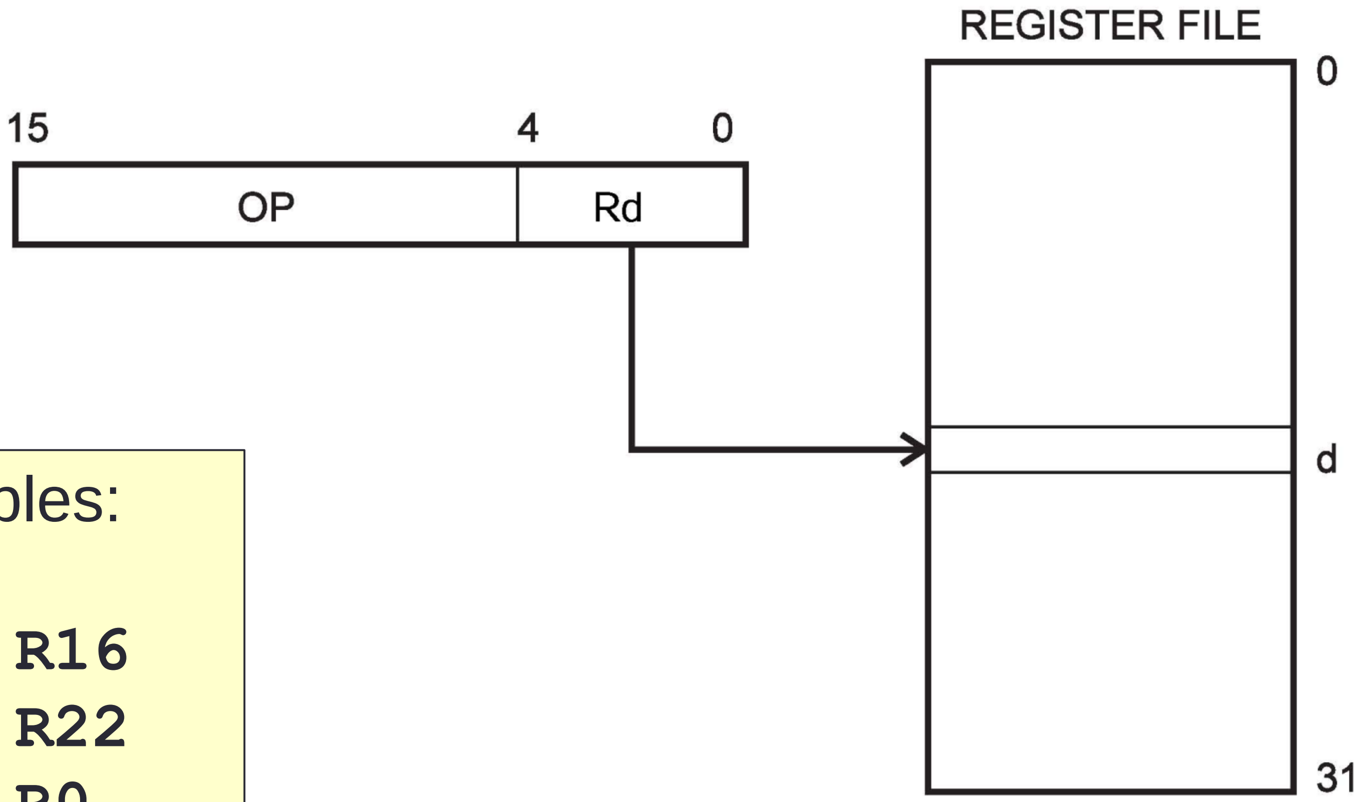
STS: 1001 001 | ddddd | 0000
AAAA AAAA AAAA AAAA

Instruction Types and Addressing Modes

- Instruction type:
  - Data transfer (**`MOV`**)
  - Arithmetic (**`ADD, SUB`**)
  - Logical
  - Program control (**`BRNE, BRCC`**)
  - I/O (**`IN, OUT`**)
- Addressing mode:
  - Register direct
  - I/O direct
  - Data direct
  - Data indirect
  - Relative program addressing

Register Direct – 1 Register

- 16-bit

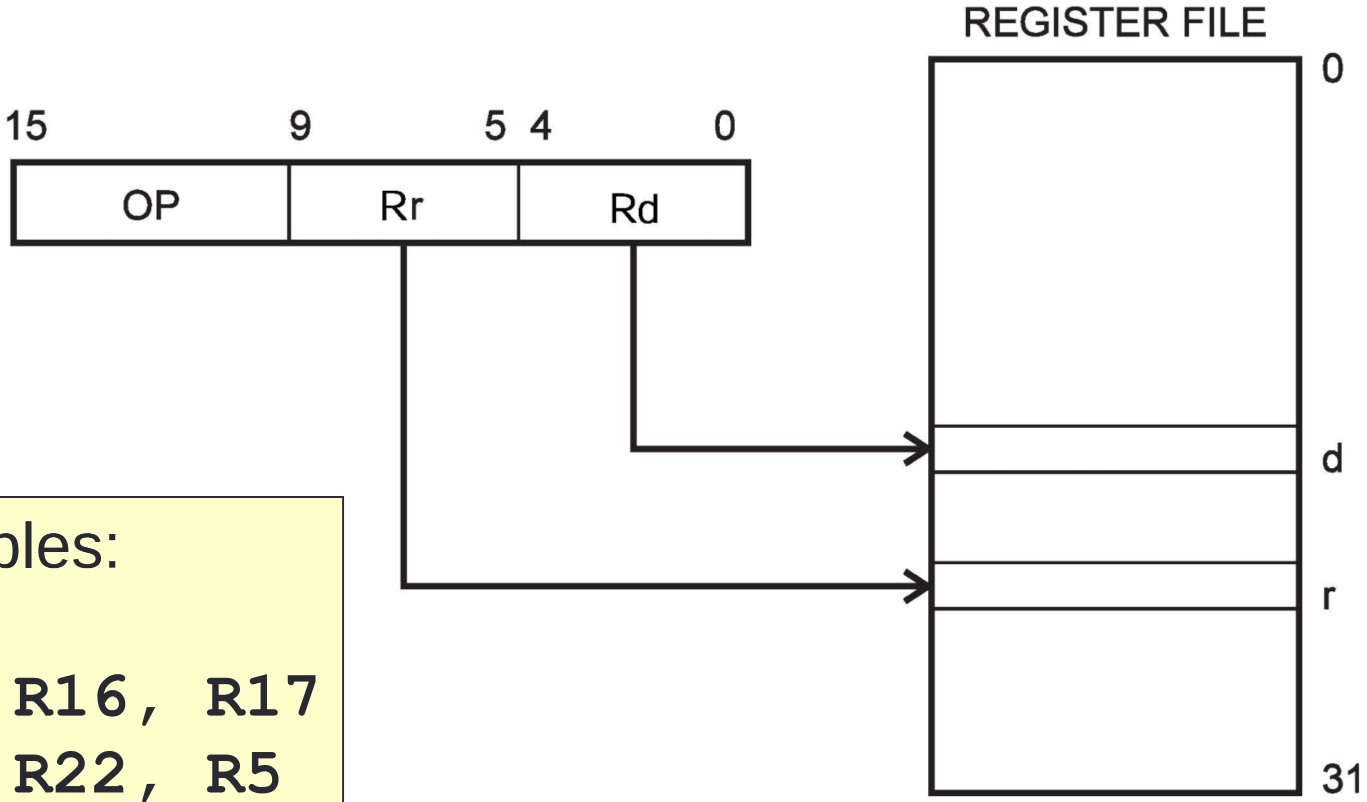


Examples:

```
INC   R16
CLR   R22
EOR   R0
```

Register Direct – 2 Registers

- 16-bit

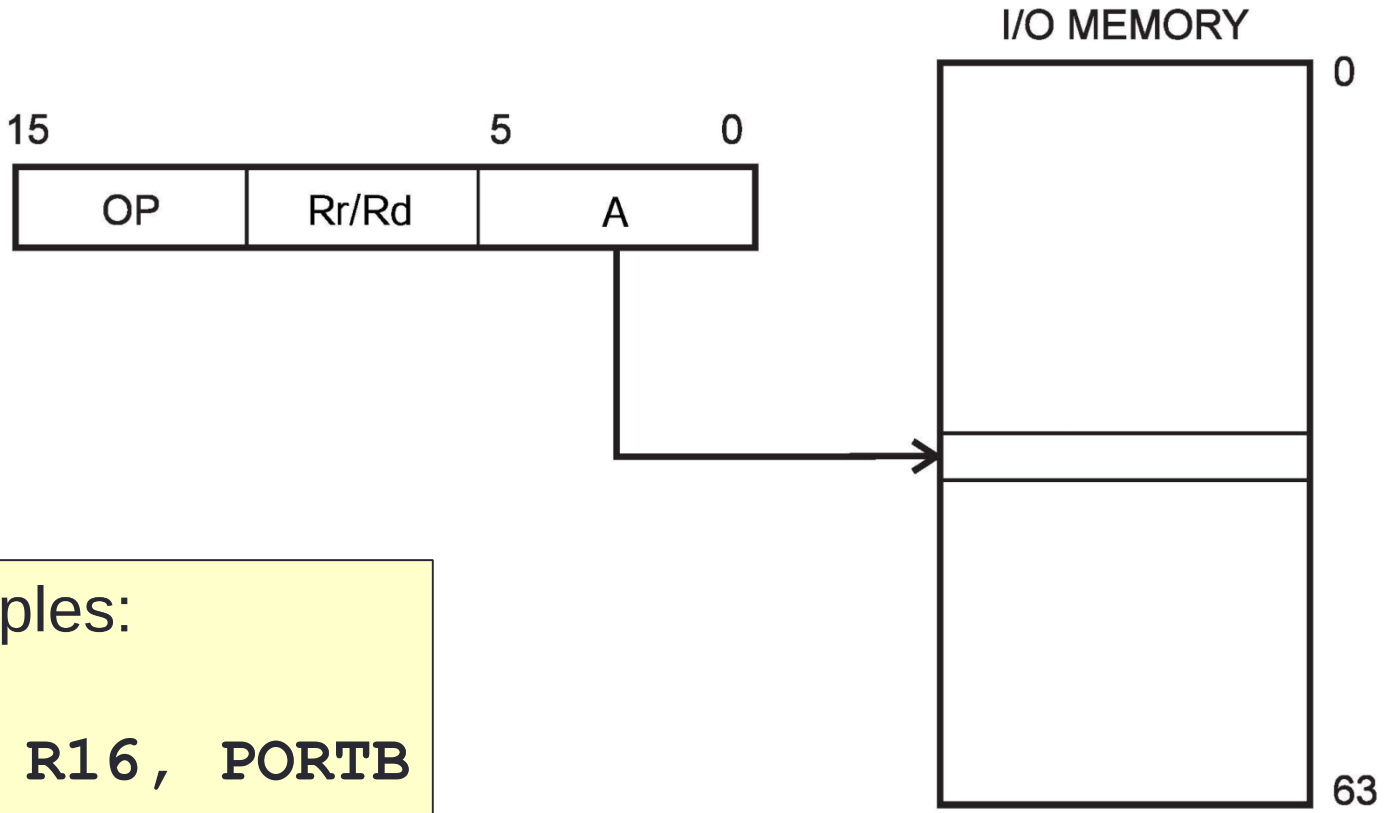


Examples:

```
ADD   R16, R17
CP    R22, R5
MOV   R0, R1
```

I/O Direct

- 16-bit

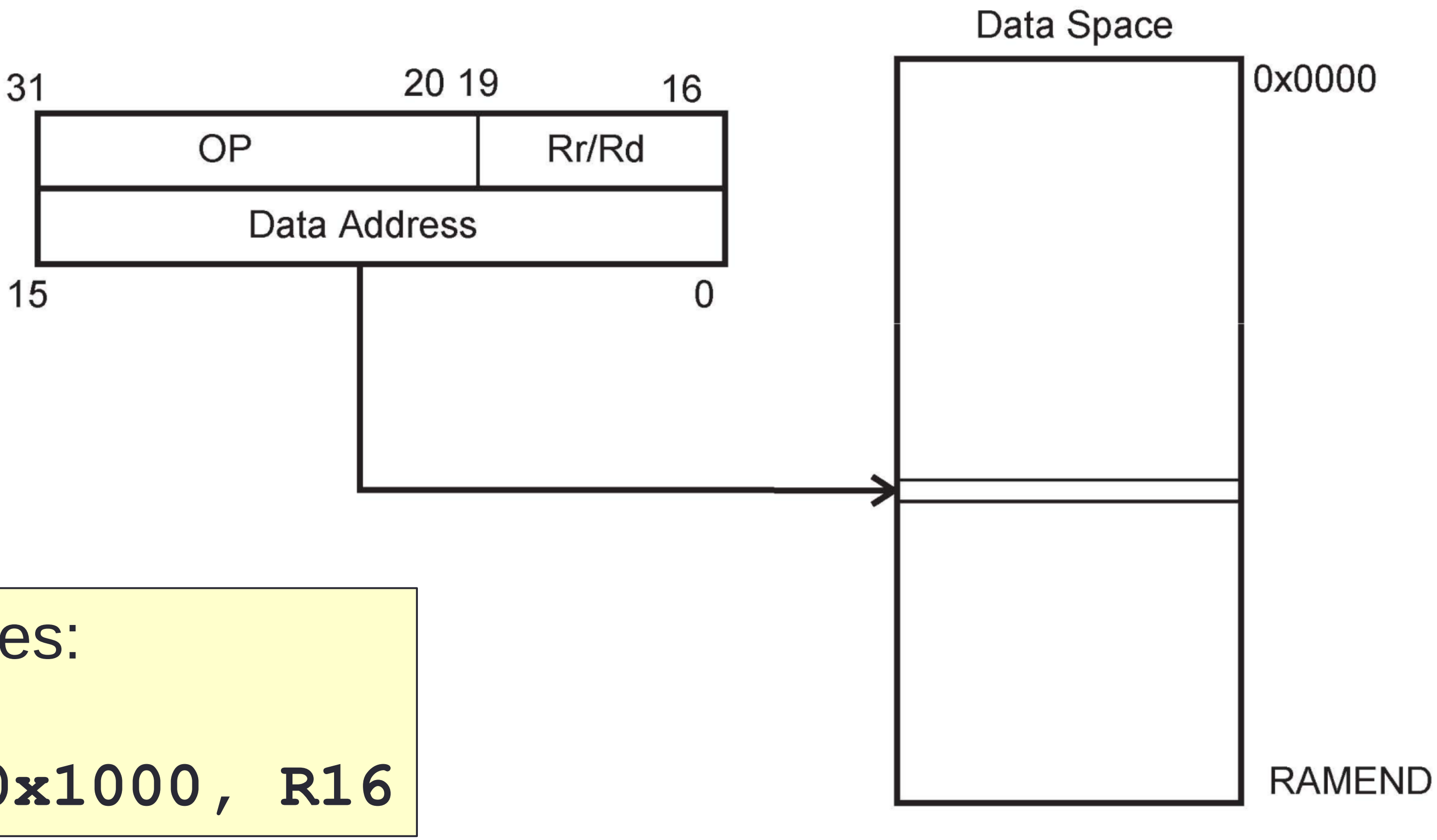


Examples:

```
IN   R16, PORTB
OUT  PORTC, R20
```

Data Direct

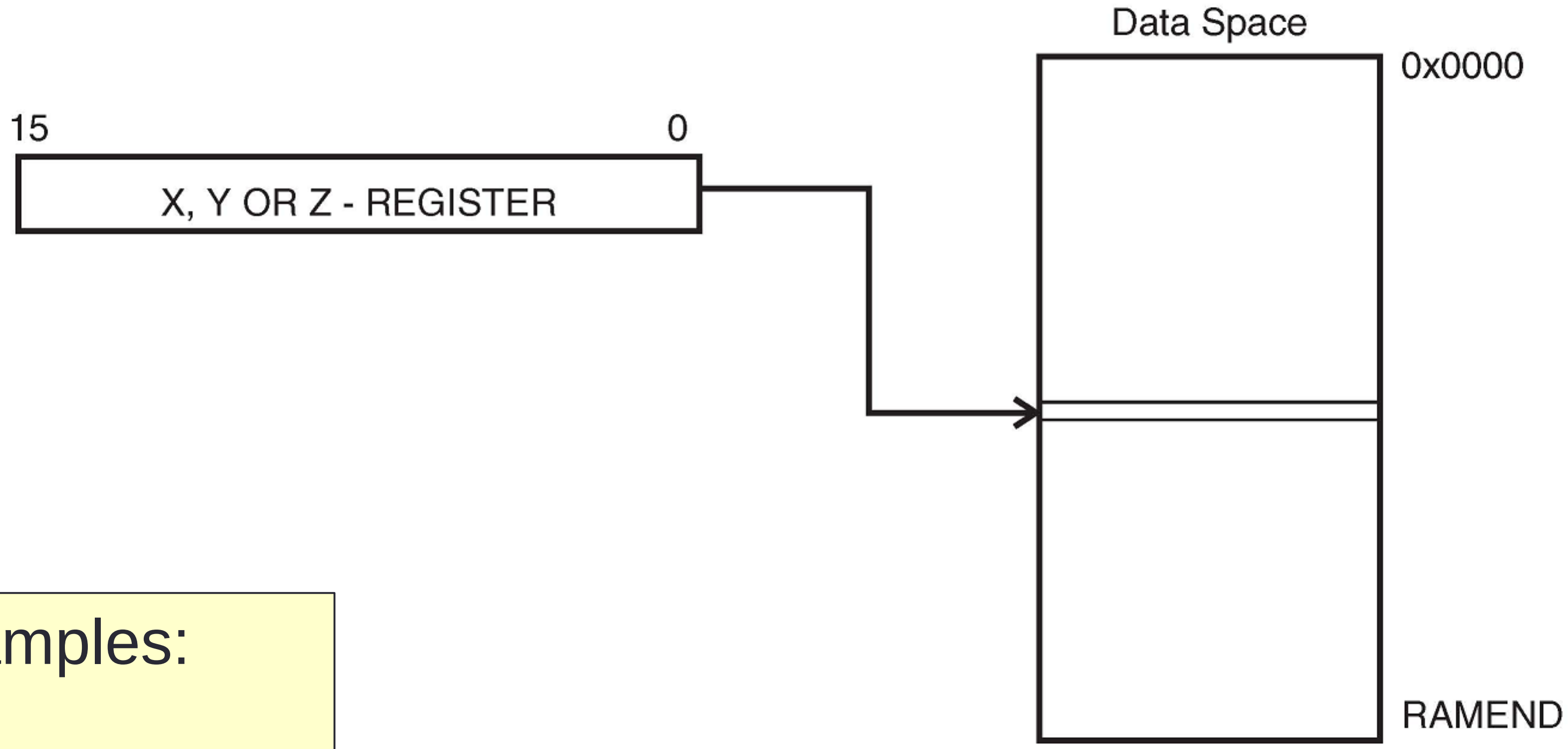
- 32-bit



Examples:

```
STS  0x1000, R16
```

Data Indirect

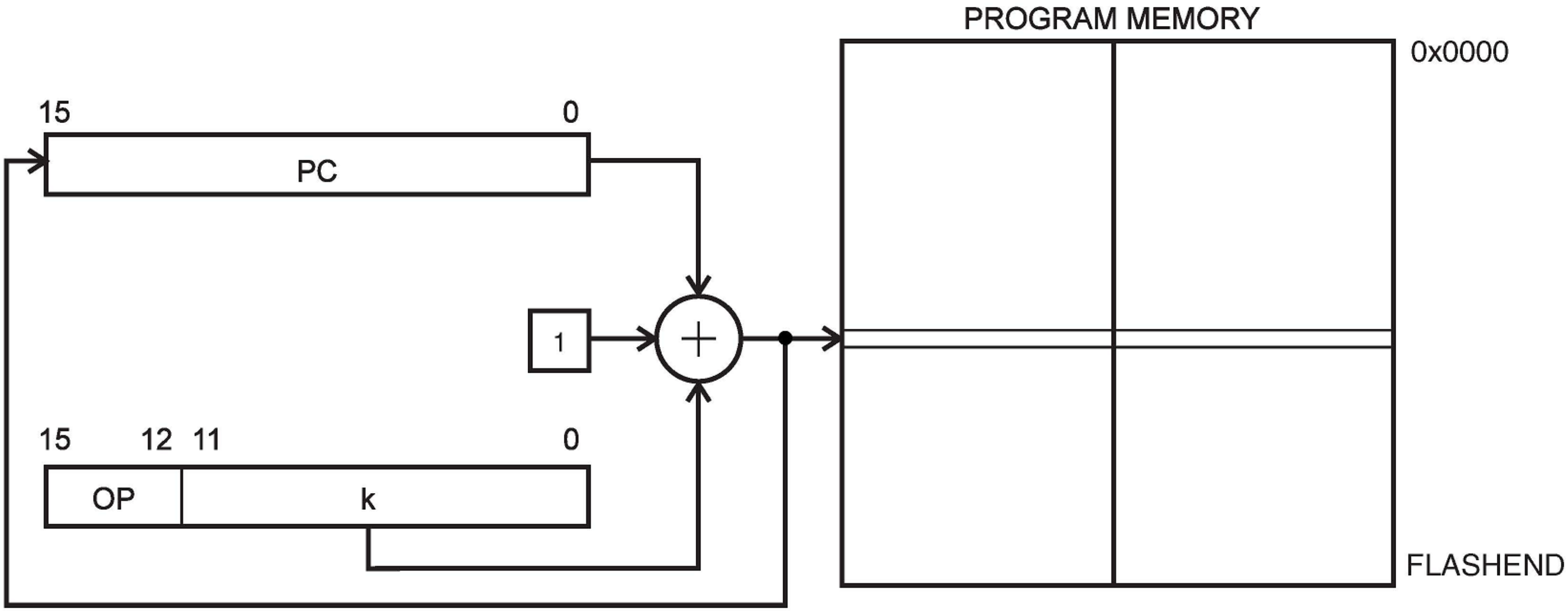


Examples:

```
LD    R16, Y
ST    Z, R16
```

Relative Program Addressing

- Program counter (PC) is involved



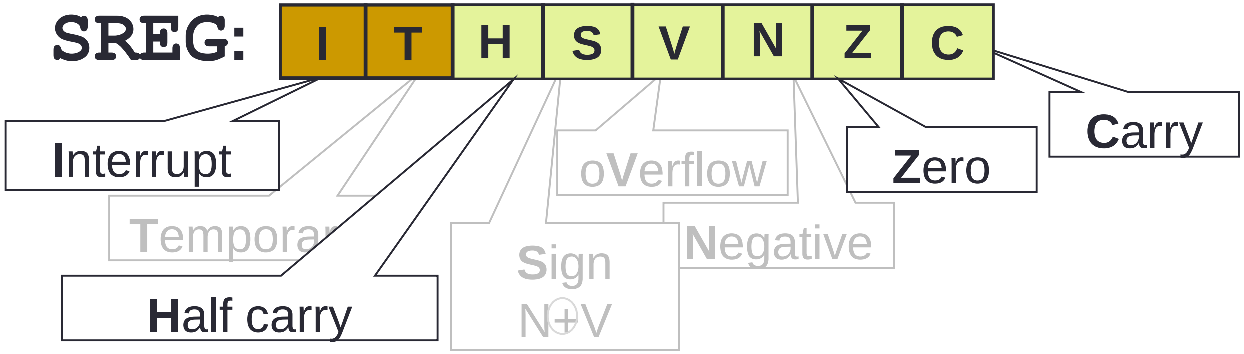
Examples:

RJMP

Outline

- Accessing data memory
- AVR assembly language
  - Structure of AVR assembly
  - Machine code
- Status register
- Getting started

Status Register (SREG)



SREG: I T H S V N Z C

Interrupt

Temporar

Half carry

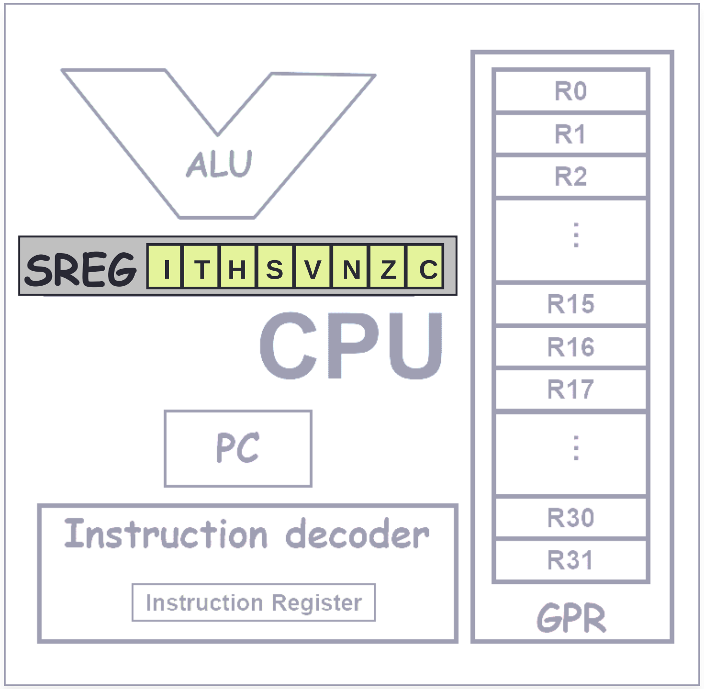
Sign
$N \oplus V$

oVerflow

Negative

Zero

Carry



Status Flags

- Information regarding “arithmetic calculation” results
- Altered automatically by the ALU after each instruction executed
- Used for a subsequent conditional jump instruction

GPR:	D7	D6	D5	D4	D3	D2	D1	D0

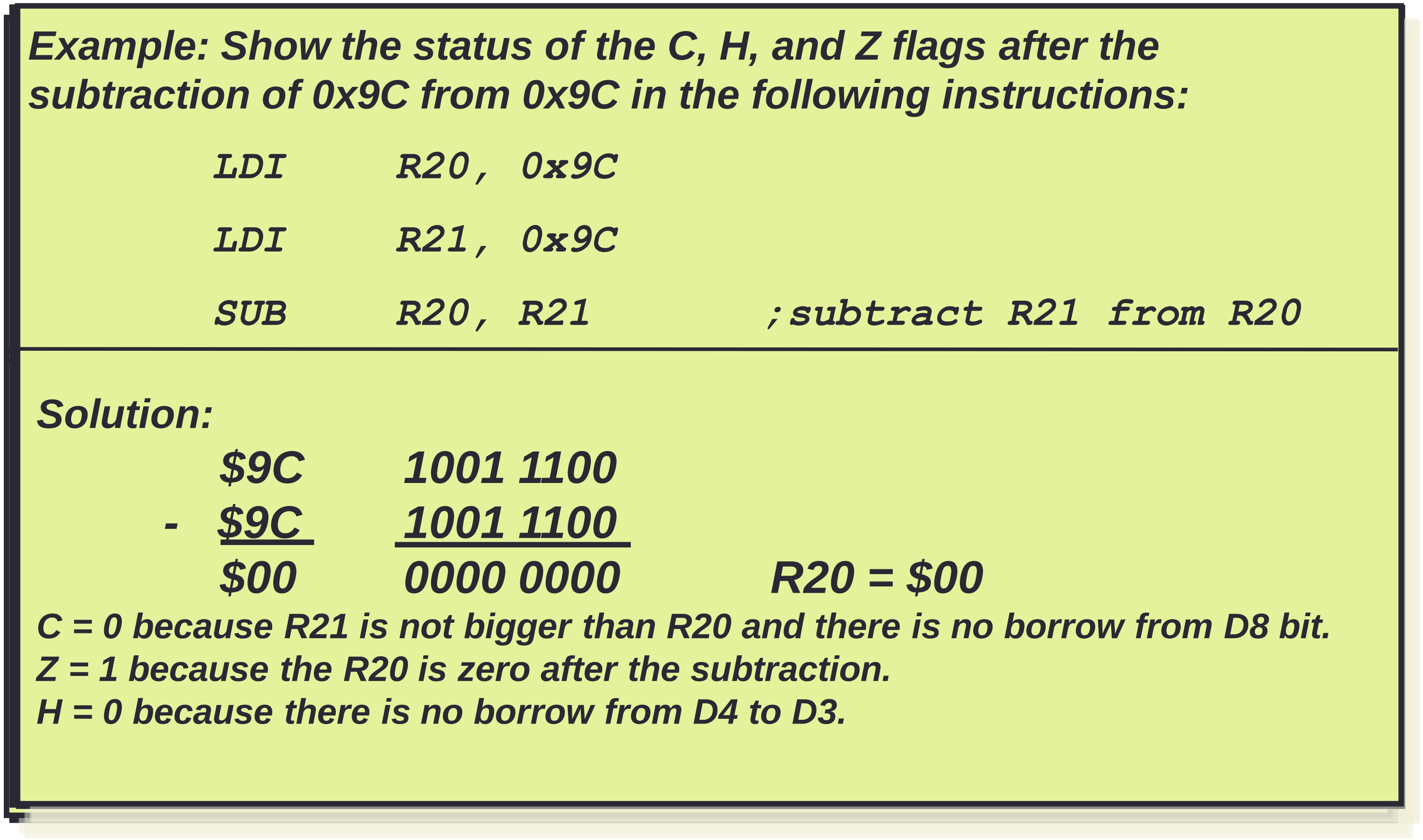
Zero:

Carry:

Half carry:

C, H, and Z Flags

- Carry, half-carry, and zero flags



Example: Show the status of the C, H, and Z flags after the subtraction of 0x9C from 0x9C in the following instructions:

```
LDI     R20, 0x9C
LDI     R21, 0x9C
SUB     R20, R21        ;subtract R21 from R20
```

Solution:

```
    $9C     1001 1100
 -  $9C     1001 1100
    $00     0000 0000        R20 = $00
```

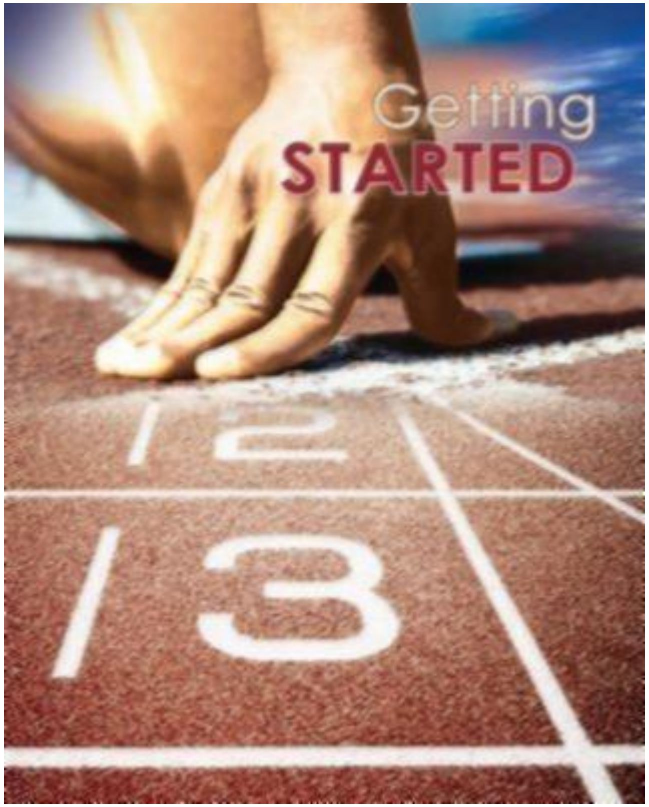
C = 0 because R21 is not bigger than R20 and there is no borrow from D8 bit.

Z = 1 because the R20 is zero after the subtraction.

H = 0 because there is no borrow from D4 to D3.

Outline (Cont'd)

- Accessing data memory
- AVR assembly language
  - Structure of AVR assembly
  - Machine code
- Status register
- Getting started



Reference

- ATmega328P datasheet
- AVR 8-bit instruction set
- AVR1022: assembler user guide
- M. A. Mazidi, S. Naimi, and S. Naimi, *The AVR Microcontroller and Embedded Systems: Using Assembly and C*, Prentice Hall, 2010
- AVR GCC library help http://nongnu.org/avr-libc/user-manual/modules.html