Common Special-Purpose Registers

- Program counter (PC): address of memory from which the next instruction will be fetched
- Status register (SR): stores basic state of the processor
- Instruction register: stores the recently fetched instruction
Atmel Mega8

Status register

• Many machine instructions affect the state of this register
Mega8 Status Register

- **Interrupt enable**
  - If ‘1’, the currently executing program can be interrupted by another event (e.g., a byte arriving through the serial port)
Mega8 Status Register

Half carry flag

- Set if an arithmetic operation resulted in a carry from the first nybble to the next
Two’s complement overflow flag
• Set if an arithmetic operation resulted in an overflow in two’s complement (e.g., incrementing an 8-bit number whose value is 127)
Mega8 Status Register

Negative flag

- Set if an arithmetic operation resulted in a negative value
Mega8 Status Register

Zero flag
- Set if an arithmetic operation resulted in a value of zero
Mega8 Status Register

Carry flag
- Set if an arithmetic operation resulted in a carry (with an unsigned value)
Some Mega8 Memory Operations

**LDS Rd, k**
- Load SRAM memory location k into register Rd
- Rd <- (k)

**STS Rd, k**
- Store value of Rd into SRAM location k
- (k) <- Rd

We refer to this as “Assembly Language”
Some Mega8 Memory Operations

**LD Rd, Ry**
- Load SRAM memory location indicated by Ry into register Rd
- Rd <- (Ry)

**ST Rd, Ry**
- Store value of Rd into SRAM location indicated by the value of Ry
- (Ry) <- Rd
Some Mega8 Arithmetic and Logical Instructions

**ADD Rd, Rr**
- Rd and Rr are registers
- Operation: Rd ← Rd + Rr
- Also affects status register (zero, carry, etc.)

**ADC Rd, Rr**
- Add with carry
- Rd ← Rd + Rr + C
Some Mega8 Arithmetic and Logical Instructions

**NEG Rd**: take the two’s complement of Rd

**AND Rd, Rr**: bit-wise AND with a register

**ANDI Rd, K**: bit-wise AND with a constant

**EOR Rd, Rr**: bit-wise XOR

**INC Rd**: increment Rd

**MUL Rd, Rr**: multiply Rd and Rr (unsigned)

**MULS Rd, Rd**: multiply (signed)
Some Mega8 Test Instructions

**CP Rd, Rr**
- Compare Rd with Rr
- Alters the status register

**TST Rd**
- Test for zero or minus
- Alters the status register
Some Program Flow Instructions

**RJMP k**
- Change the program counter by \( k + 1 \)
- \( PC \leftarrow PC + k + 1 \)

**BRCS k**
- Branch if carry set
- If \( C==1 \) then \( PC \leftarrow PC + k + 1 \)
Connecting Assembly Language to C

• Our C compiler is responsible for translating our code into Assembly Language

• Today, we rarely program in Assembly Language
  – Embedded systems are a common exception
  – Also: it is useful in some cases to view the assembly code generated by the compiler
An Example

A C code snippet:

if(A > B) {
    D = D + A;
}

An Example

A C code snippet:

```c
if(A > B) {
    D = D + A;
}
```

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRSH 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

........
An Example

A C code snippet:

```c
if(A > B) {
    D = D + A;
}
```

Load the contents of memory location A into register 1

The Assembly:

```assembly
LDS R1 (A)  
LDS R2 (B)  
CP R2, R1   
BRSH 3      
LDS R3 (D)  
ADD R3, R1  
STS (D), R3 
```

……..
Atmel Mega8

LDS R1 (A)
An Example

A C code snippet:

```c
if(A > B) {
    D = D + A;
}
```

The Assembly:

```assembly
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRSH 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

Load the contents of memory location B into register 2

PC
An Example

A C code snippet:

```c
if(A > B) {
    D = D + A;
}
```

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRSH 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

Compare the contents of register 2 with those of register 1.
This results in a change to the status register.

PC
Atmel Mega8

CP R2, R1
An Example

A C code snippet:

```c
if(A > B) {
    D = D + A;
}
```

Branch if greater than or equal to will jump ahead 3 instructions if true

The Assembly:

```assembly
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRSH 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
........
```
Atmel Mega8

BRSH
An Example

A C code snippet:

```c
if (A > B) {
    D = D + A;
}
```

Branch if greater than or equal to will jump ahead 3 instructions if true

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRSH 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

if true

PC
An Example

A C code snippet:

```c
if(A > B) {
    D = D + A;
}
```

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRSH 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

Not true: execute the next instruction
An Example

A C code snippet:

```c
if(A > B) {
    D = D + A;
}
```

Load the contents of memory location D into register 3

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRSH 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

PC
An Example

A C code snippet:

```c
if(A > B) {
    D = D + A;
}
```

Add the values in registers 1 and 3 and store the result in register 3

The Assembly:

```assembly
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRSH 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

PC
Atmel Mega8

ADD R3, R1
An Example

A C code snippet:

```
if(A > B) {
    D = D + A;
}
```

Store the value in register 3 back to memory location D

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRSH 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

...........
Atmel Mega8

STS (D), R3
Next Week

• Interrupts
• Analog I/O