EE-222: Microprocessor Systems

AVR Programming in C

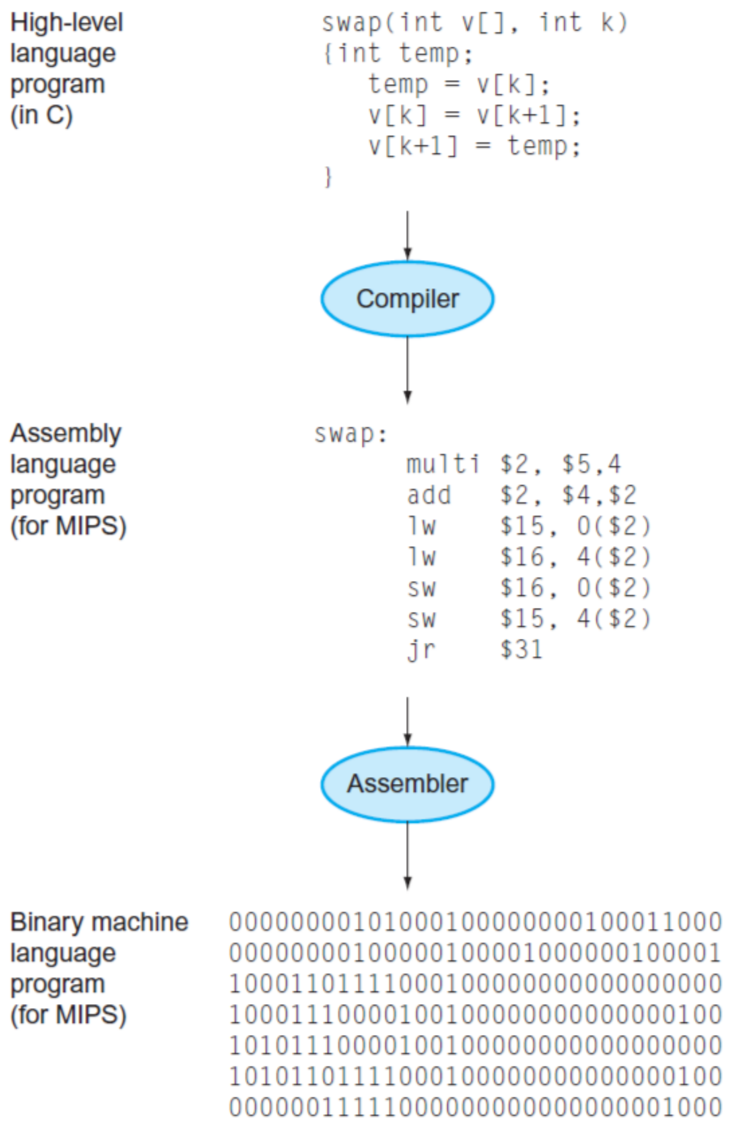
Instructor: Dr. Arbab Latif

NUST
SCHOOL OF ELECTRICAL ENGINEERING &
COMPUTER SCIENCE (SEECS)

Languages

- High Level Languages
  - Easy to develop and update
  - Acceptable performance
  - Portable

- Low Level Languages
  - High performance
  - Not portable


High-level language program (in C)
Compiler
Assembly language program (for MIPS)
Assembler
Binary machine language program (for MIPS)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

```
swap:
      multi $2, $5,4
      add   $2, $4,$2
      lw    $15, 0($2)
      lw    $16, 4($2)
      sw    $16, 0($2)
      sw    $15, 4($2)
      jr    $31
```

```
00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

Figure taken from Computer Organization and Design (COD) 5th Ed.

A simple C program

- Write a program that calculate the sum of {1,3,…,13,15}

```
int main ()
{
    unsigned int sum;

    for (int i = 1; i <= 15; i+=2)
        sum += i;

    while (1);
    return 0;
}
```

I/O Programming in C

I/O Programming in C

- To access a PORT register as a byte:
  - Use the PORTx label, x=Port name i.e PORTB

- Data Direction is accessed using:
  - DDRx

- PIN register is accessed using:
  - PINx

Accessing I/O registers

- Example 1: Write an AVR C program to send value 0xAA to PORTD.

```
#include <avr/io.h>

int main ()
{
   DDRD = 0xFF;
   PORTD = 0xAA;

   while (1);
   return 0;
}
```

Accessing I/O registers

- Example 2: Write an AVR C program to calculate PINB + PINC and send the result to PORTD.

```
#include <avr/io.h>

int main ()
{
   DDRB = 0x00;
   DDRC = 0x00;
   DDRD = 0xFF;

   while (1)
      PORTD = PINB + PINC;
   return 0;
}
```

What is happening in Machine Language?

```
int Fibonnaci (int n);

int main ()
{
   unsigned int sum = 0;

   for(int j = 0; j < 11; j++)
   {
        sum += Fibonnaci (j);
   }

   while(1);
   return 0;
}
```

```
int Fibonnaci (int n)
{
   int a1 = 1;
   int a2 = 1;
   int i;
   int temp;

   if(n == 0)
        return 0;
   else
        if(n <= 2)
                return 1;
   else{
        for(i = 2; i < n; i++)
        {
                temp = a2;
                a2 = a1 + a2;
                a1 = temp;

        }
        return a2;
   }
}
```

AVR C Datatypes

Data Types

- Use unsigned whenever you can
- unsigned char instead of unsigned int if you can

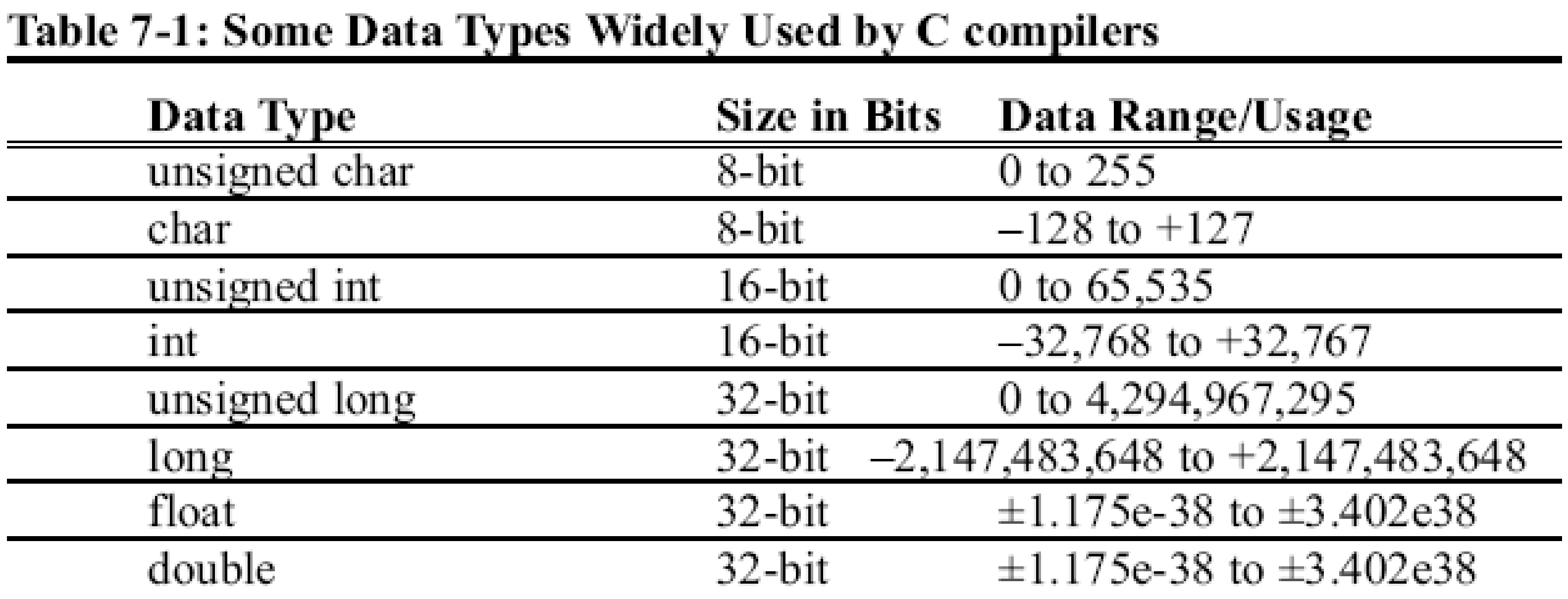
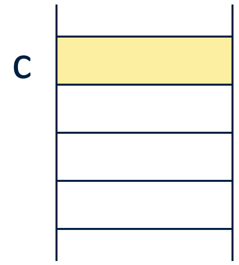


Table 7-1: Some Data Types Widely Used by C compilers

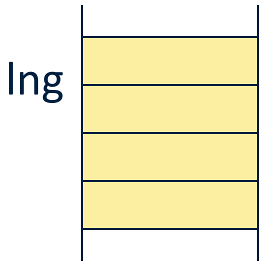
Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
char	8-bit	–128 to +127
unsigned int	16-bit	0 to 65,535
int	16-bit	–32,768 to +32,767
unsigned long	32-bit	0 to 4,294,967,295
long	32-bit	–2,147,483,648 to +2,147,483,648
float	32-bit	±1.175e-38 to ±3.402e38
double	32-bit	±1.175e-38 to ±3.402e38

Data types (cont.)

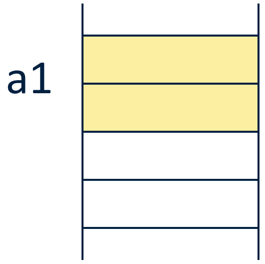
- char c;



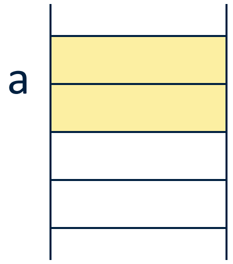
- long lng;



- int a1;



- unsigned int a;



Unsigned int is used to define 16-bit variables:

i.e counter values of more than 256 etc

Takes two bytes in RAM

Avoid using int data type unless you have to

int vs. unsigned int

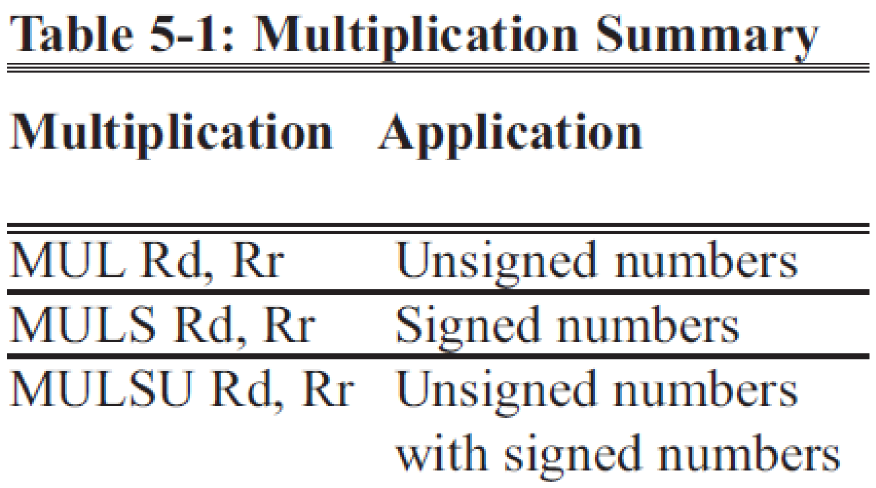


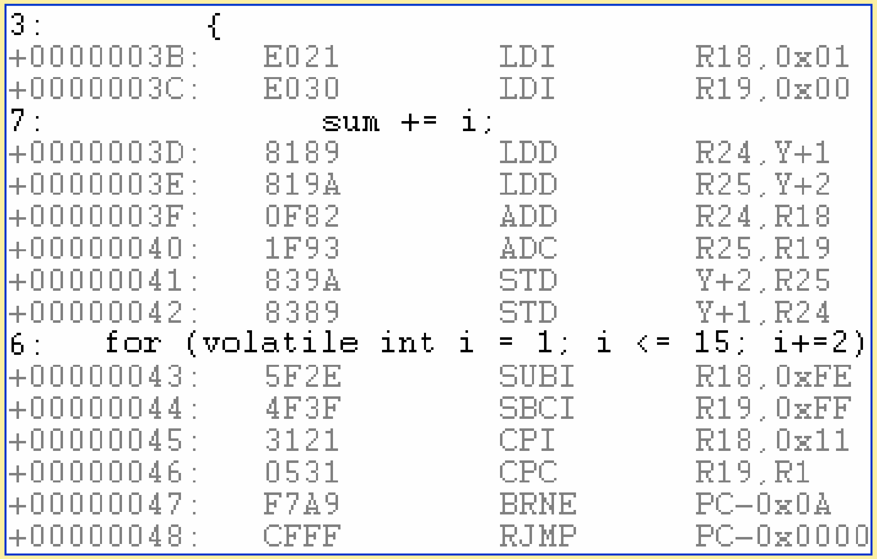
Table 5-1: Multiplication Summary

Multiplication	Application
MUL Rd, Rr	Unsigned numbers
MULS Rd, Rr	Signed numbers
MULSU Rd, Rr	Unsigned numbers with signed numbers

- int a1 = 5;
  int a2 = 3;
  int a3 = 9;
  b = (a1 * a2) + a3;

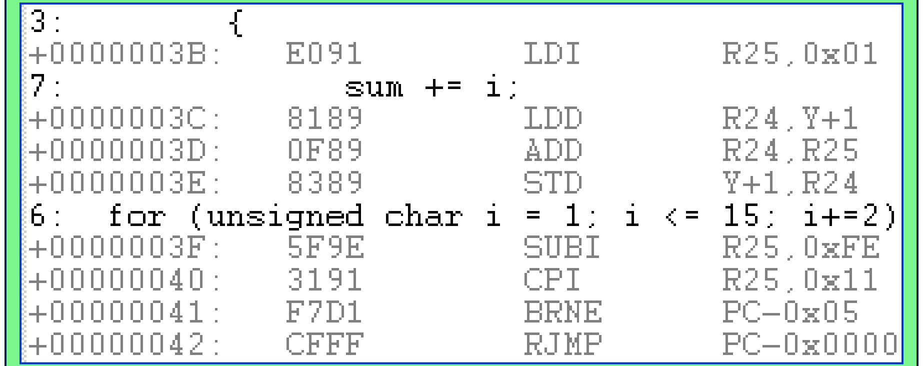
Choosing optimized data type

```
unsigned int sum;

for (int i = 1; i <= 15; i+=2)
   sum += i;
```



```
3:              {
+0000003B:    E021          LDI         R18,0x01
+0000003C:    E030          LDI         R19,0x00
7:                 sum += i;
+0000003D:    8189          LDD         R24,Y+1
+0000003E:    819A          LDD         R25,Y+2
+0000003F:    0F82          ADD         R24,R18
+00000040:    1F93          ADC         R25,R19
+00000041:    839A          STD         Y+2,R25
+00000042:    8389          STD         Y+1,R24
6:    for (volatile int i = 1; i <= 15; i+=2)
+00000043:    5F2E          SUBI        R18,0xFE
+00000044:    4F3F          SBCI        R19,0xFF
+00000045:    3121          CPI         R18,0x11
+00000046:    0531          CPC         R19,R1
+00000047:    F7A9          BRNE        PC-0x0A
+00000048:    CFFF          RJMP        PC-0x0000
```

```
unsigned char sum;

for (char i = 1; i <= 15; i+=2)
   sum += i;
```



```
3:              {
+0000003B:    E091          LDI         R25,0x01
7:                 sum += i;
+0000003C:    8189          LDD         R24,Y+1
+0000003D:    0F89          ADD         R24,R25
+0000003E:    8389          STD         Y+1,R24
6:    for (unsigned char i = 1; i <= 15; i+=2)
+0000003F:    5F9E          SUBI        R25,0xFE
+00000040:    3191          CPI         R25,0x11
+00000041:    F7D1          BRNE        PC-0x05
+00000042:    CFFF          RJMP        PC-0x0000
```

Accessing I/O registers

```
#include <avr/io.h>

int main ()
{
   DDRD = 0xFF;

  while(1)
   {
        for (unsigned char i = 0; i <= 9; i++)
             PORTD = i;
   }
   return 0;
}
```