

CprE 288 – Introduction to Embedded Systems

ATmega128 Assembly Programming: Translating C Control Statements and Function Calls

Instructors:

Dr. Phillip Jones (Sections F, G, J)

Dr. Zhao Zhang (Sections A, B, C, D, E)

Major Classes of Assembly Instructions

- Data Movement
 - Move data between registers
 - Move data in & out of SRAM
 - Different addressing modes
- Logic & Arithmetic
 - Addition, subtraction, etc.
 - AND, OR, bit shift, etc.
- **Control Flow**
 - Control which sections of code should be executed (e.g. In C “IF”, “CASE”, “WHILE”, etc.
 - Typically the result of Logic & Arithmetic instructions help decided what path to take through the code.

DO-WHILE Loop

do

do-body;

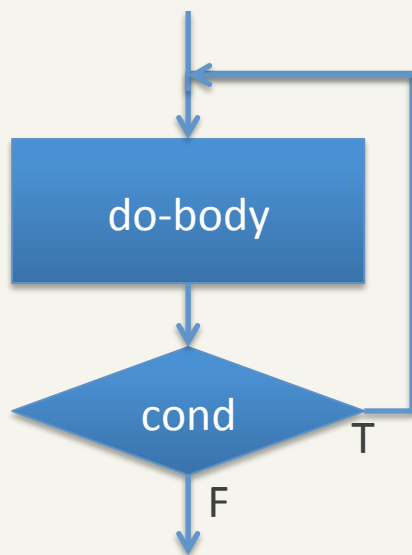
while (*cond*);

Example:

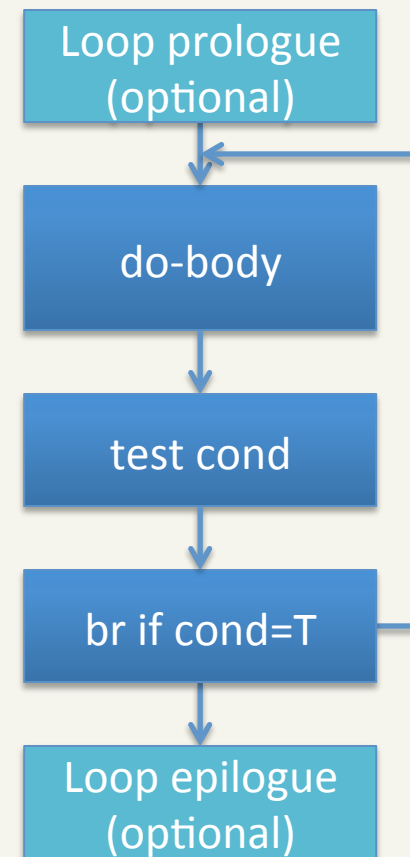
```
void strcpy (char *dst,  
             char *src)  
{  
    char ch;  
    do {  
        ch = *src++;  
        *dst++ = ch;  
    } while (ch);  
}
```

DO-WHILE Loop

Control and Data Flow Graph



Linear Code Layout



DO-WHILE Loop

```
; parameter: dst=>R25:R24, src=>R23:R22  
; reg use: dst=>Z-reg, src=>X-reg  
strcpy:
```

```
    movw r30, r24
```

```
    movw r26, r22
```

```
loop:
```

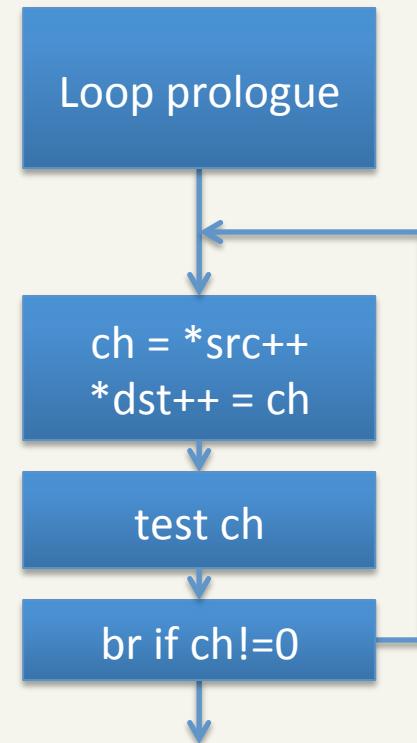
```
    ld    r20, X+
```

```
    st    Z+, r20
```

```
    tst   r20
```

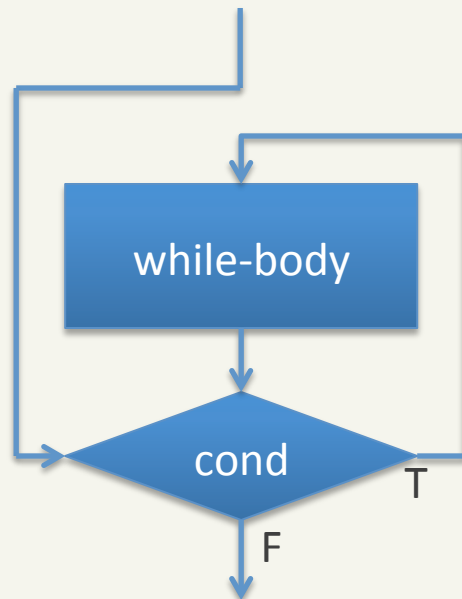
```
    brne  loop
```

```
    ret
```

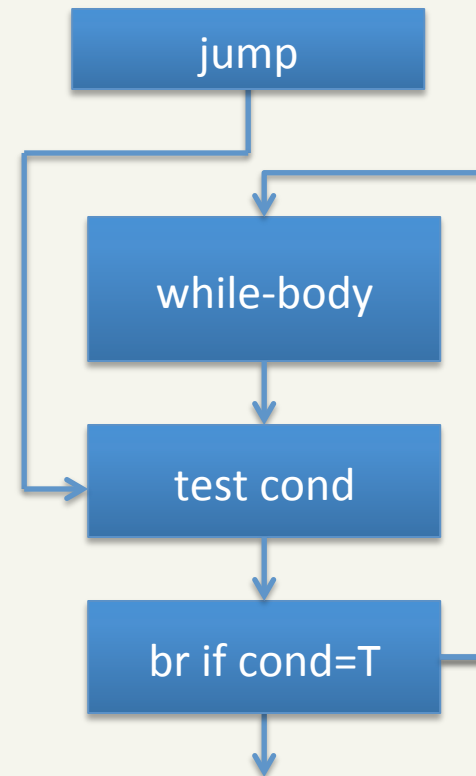


WHILE Loop

Control and Data Flow Graph



Linear Code Layout



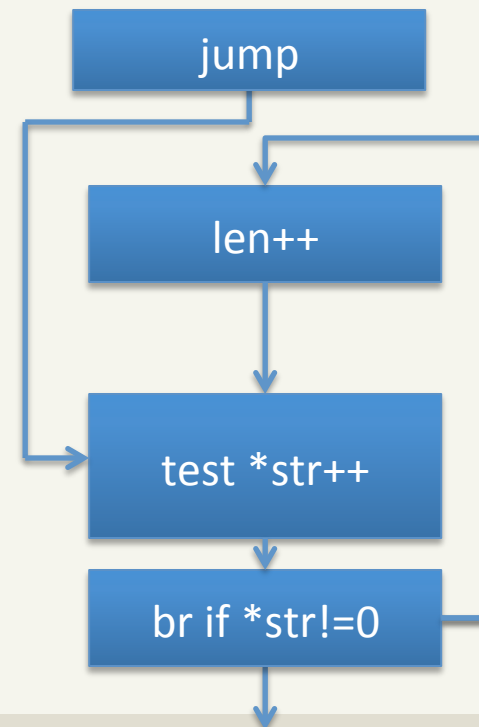
(optional prologue and epilogue not shown)

strlen(): return the length of a C string

```
int strlen(char *str)
{
    int len = 0;
    while (*str++)
        len++;
    return len;
}
```

WHILE Loop

```
; parameter: str=>r25:r24
; reg use: str=>Z-reg, len=>r25:r24, ch=>r22
strcpy:
    movw r30, r24
    clr  r24                ; len = 0
    clr  r25
    rjmp test
loop:
    adiw r24, 1
test:
    ld    r22, Z+
    tst   r22
    brne loop
    ret
```



FOR Loop

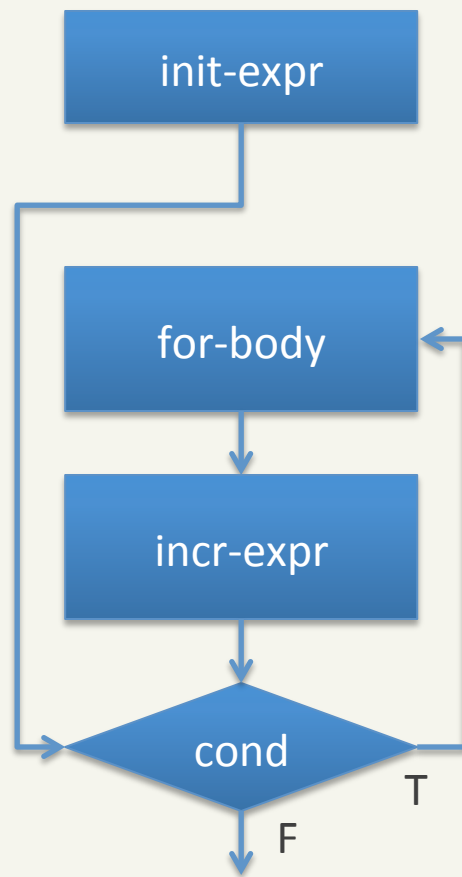
*for (init-expr; cond-expr; incr-expr)
for-body;*

Example:

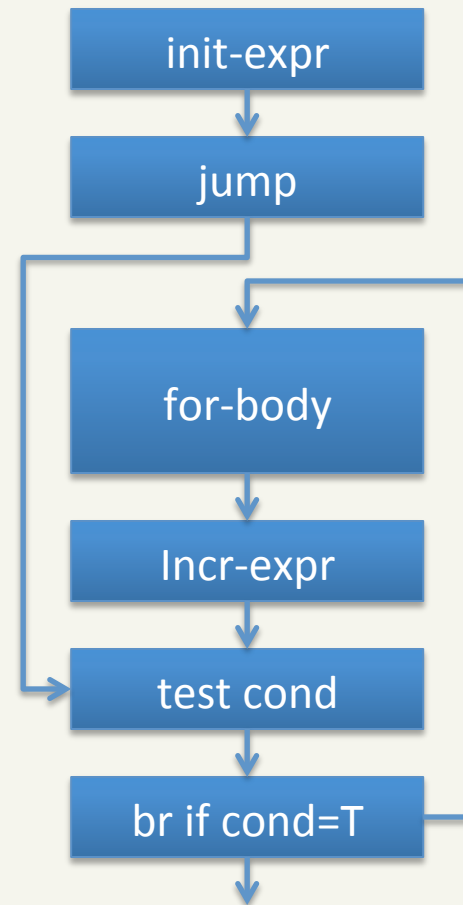
```
unsigned char checksum(unsigned char data[],  
                        int N)  
{  
    unsigned char checksum = 0;  
    for (int i=N; i>=0; i--)  
        checksum ^= data[i];  
    return checksum;  
}
```

FOR Loop

Control and Data Flow Graph



Linear Code Layout



(optional prologue and epilogue not shown)

FOR Loop

```
; parameter: A=>r25:r24, N=>r23:r22
```

```
; reg use: A=>Z-reg, checksum=>r24, i=>r27:r26, ch=>r20
```

checksum:

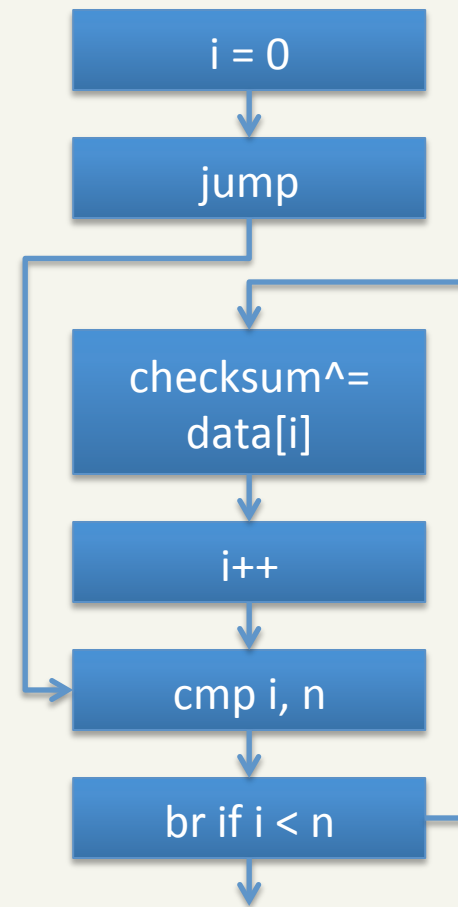
```
movw    r30, r24
```

clr r24

clr r26

```
clr    r27
```

rjmp cond



FOR Loop

; parameter: A=>r25:r24, N=>r23:r22

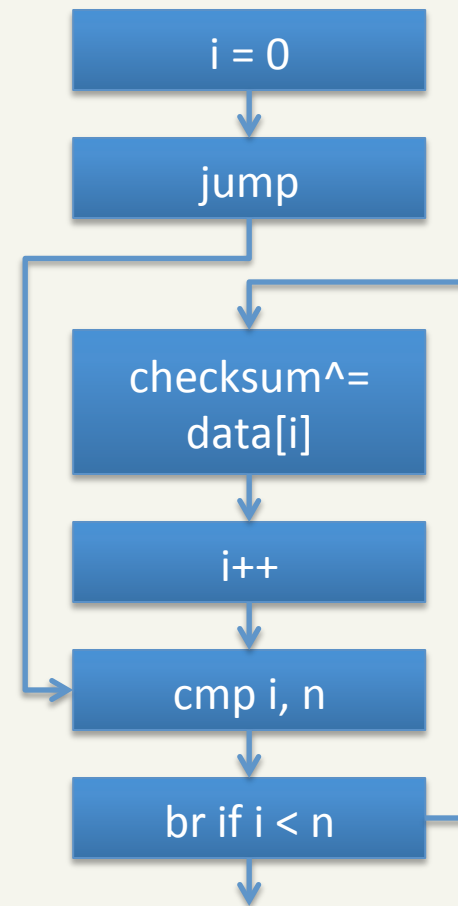
; reg use: A=>Z-reg, checksum=>r24, i=>r27:r26, ch=>r20

loop:

```
ld    r20, Z+  
xor   r24, r20  
adiw  r26, 1
```

cond:

```
cp    r26, r22  
cpc   r27, r23  
brne  loop  
clr   r25  
ret
```



FOR Loop

Another example

```
extern int data[];  
  
// clear the first n elements of data[]  
void clear_data(int n)  
{  
    for (int i = 0; i < n; i++)  
        data[i] = 0;  
}
```

FOR Loop Example

```
; parameter: n=>r25:r24
; reg use: i=>r26:r27,
;           &data[i]=>r31:r30 (Z-reg)
clear_data:
    ldi    r30, lo8(data)        ; Z-reg = data
    ldi    r31, hi8(data)
    clr    r26                    ; i = 0
    clr    r27
    rjmp   cond_check            ; jump to condition
```

FOR Loop Example

```
for_loop:
    st    Z+,    r1           ; data[i] = 0
    st    Z+,    r1
    adiw  r26, 1             ; i++
cond_check:
    cp    r26, r24           ; cmp i, n
    cpc   r27, r25
    brlt  for_loop          ; br if i<n
    ret
```

FOR Loop Example: Optimized Version

```
; n=>r25:r24, &data[i]>r31:r30 (Z-reg)
clear_data:
    ldi    r30, lo8(data) ; Z-reg = data
    ldi    r31, hi8(data)
    cp     r24, r1         ; test condition for 1st time
    cpc    r25, r1
    rjmp   cond_check     ; jump to condition
for_loop:
    st     Z+, r1          ; data[i] = 0
    st     Z+, r1
    sbiw   r24, 1          ; n--
cond_check:
    brne   for_loop       ; br if n!=0
    ret
```