Delay Loops
Program Loops

• Many applications require the program to do the same operations over and over
• An “endless loop” (or infinite loop) is a sequence of instructions that the computer will execute forever

Example:

; This program module continually
; reads from Port T, increments the
; value, and writes it back to Port T
Loop ldaa PTT ; Read port T
inca ; increment A
staa PTT ; store to PTT
bra Loop ; repeat

Figure 2.4 An infinite loop
Finite Loops

• A “finite loop” is a sequence of instructions that will be executed a finite number of times

• We need to test some condition that will tell the computer when to stop

![Diagram of the While ... Do looping construct](image1)

Figure 2.6 The While ... Do looping construct

![Diagram of the Repeat ... Until looping construct](image2)

Figure 2.7 The Repeat ... Until looping construct
Loops

- Example: execute a block of code 500 times

![Flowchart](image)

Figure 2.7 The Repeat ... Until looping construct

**Pseudocode**

Load a register with the value 500
do
    (block of code)
    decrement the register
while (register ≠ zero)
Assembly Code for Loop

<table>
<thead>
<tr>
<th>ldd</th>
<th>#500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop</td>
<td>(block of code)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>subd</td>
<td>#1</td>
</tr>
<tr>
<td>bne</td>
<td>Loop</td>
</tr>
</tbody>
</table>

- Could you use other registers for the loop counter?
- What’s the maximum count?

Loops are used so often that they provide a single instruction, \( \text{dbne} \), that both decrements a register and then branches if not zero:

<table>
<thead>
<tr>
<th>ldd</th>
<th>#500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop</td>
<td>(block of code)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>dbne</td>
<td>D,Loop</td>
</tr>
</tbody>
</table>

- There are also instructions that increment instead of decrement, and branch if equal to zero
  - Full list:
    - dbeq
    - dbne
    - ibeq
    - ibne
Time to execute a loop

- The E-clock in our MCU is 24 MHz
- Period = 1/24 MHz = 0.0416 usec (microseconds)

Example loop:

```assembly
ldd #500 ; 2 cycles
loop: nop ; 1 cycle
subd #1 ; 2 cycles
bne loop ; 3/1 cycles
```

“nop” – means no operation; this instruction just delays a little

bne takes 3 cycles if branch is taken; 1 if not

- Total time (not counting initial ldd):

  $500 \times (1+2+3) = 3000$ cycles

  So $3000 \times 0.0416$ usec = 125 usec

Actually, for the last iteration bne only takes 1 cycle instead of 3
Delay loops

- Sometimes we want to delay a certain amount of time - we can use a loop
  - We first create a sequence of instructions that does nothing, but takes a known amount of time to execute
  - Then we repeat the sequence as many times as necessary to create the desired delay

Example:

```
; Execute the following sequence
; for as many times equal to the
; value in the X register.
loop  psha       ; 2 E cycles
      pula        ; 3 E cycles
      psha       ; 2 E cycles
      pula        ; 3 E cycles
      psha       ; 2 E cycles
      pula        ; 3 E cycles
      psha       ; 2 E cycles
      pula        ; 3 E cycles
      nop         ; 1 E cycle
      dbne x,loop ; 3 E cycles
```
Delay loops (continued)

- One iteration of this loop takes 24 E clock cycles to execute

- Assume that the HCS12 has an E-clock frequency $= 24 \text{ MHz}$ and hence a clock period of $1/24 \text{ MHz} = 0.042 \text{ us (microseconds)}$

- One iteration of the loop will take $24 \times (0.042 \text{ us}) = 1.0 \text{ us}$ to execute

- So, to delay $N$ microseconds, just load register X with the value of $N$ and run the loop

Example:

```plaintext
; Execute the following sequence
; for as many times equal to the
; value in the X register.
loop  psha        ; 2 E cycles
    pula        ; 3 E cycles
    psha        ; 2 E cycles
    pula        ; 3 E cycles
    psha        ; 2 E cycles
    pula        ; 3 E cycles
    psha        ; 2 E cycles
    pula        ; 3 E cycles
    nop         ; 1 E cycle
    dbne x,loop ; 3 E cycles
```

Total E cycles for one iteration $= 24$
Write a program loop to create a delay of 100 us (microseconds).

**Solution:** A delay of 100 us can be created by repeating the previous loop 100 times.

```
ldx #100

loop    psha       ; 2 E cycles
         pula       ; 3 E cycles
         psha       ; 2 E cycles
         pula       ; 3 E cycles
         psha       ; 2 E cycles
         pula       ; 3 E cycles
         psha       ; 2 E cycles
         pula       ; 3 E cycles
         nop        ; 1 E cycle
         dbne x,loop ; 3 E cycles
```
Write an instruction sequence to create a delay of 1 second.

**Solution:** By repeating the previous instruction sequence 10,000 times, we can create a delay of 1 second.

```assembly
ldy  #10000
outloop ldx  #100
inloop  psha ; 2
    pula ; 3
    psha ; 2
    pula ; 3
    psha ; 2
    pula ; 3
    psha ; 2
    pula ; 3
    nop ; 1
dbne x,inloop ; 3
dbne y,outloop ; 3
```

- We need to run the loop 1,000,000 times to get a 1 sec delay (because $1,000,000 \times 1 \text{ us} = 1 \text{ second}$)

- But we can’t just load register X with 1,000,000 (why not?)

- So instead, enclose the loop that takes 100 us to run inside another loop, and run the outer loop 10,000 times

- $10000 \times (100 \text{ us}) = 1 \text{ sec}$
Summary

• Loops are used to execute a block of code over and over.

• To execute the block a specified number of times, increment (or decrement) a counter.

• We can delay for a specified amount of time by executing a “do nothing” loop for a certain number of iterations.