C like Pseudo Code

register

R10 = 0x100  \( ( \text{put } 0x100 \text{ in } R10) \)

\( x = 0x0 \)  \( ( \text{put } 0x0 \text{ in } x) \)

variable

\begin{align*}
\text{repeat: } & \quad x = x + R10 \\
& \quad R10 = R10 - 1 \quad (\text{or } R10-- \text{, decrease } R10 \text{ by } 1) \end{align*}

\begin{align*}
\text{if}(R10 == 0) \quad \text{then} & \\
& \quad \text{go to continue} \quad \text{label} \\
& \quad \text{else} \quad \text{label} \\
& \quad \text{go to repeat} \quad \text{label} \\
\end{align*}

loop: \quad \text{go to loop}
Flow Charts

1. Main (program entry point)
2. Initialize SP
3. Kim the Dog
   - Repeat:
     - X = X + R10
   - R10 --
   - If R10 == 0
     - Yes
     - Exit
     - Continue
   - No

Labels and Conditions:
- Label (labels on the left)
- "Yes" on the left hand corner
- "No" on the bottom corner
- Infinite Loop
- Process blocks
Conditional Instruction

\[ JZ: \text{Jump if zero} \]

\[ \text{dec}\cdot w \ R10 \rightarrow \text{sets the Z status bit in SR to 1} \]
\[ \text{if R10 is zero, after this instruction executes} \]
\[ \text{jump to continue} \]
\[ \text{if Z bit in SR is set, otherwise continue from here} \]

\[ \text{repeat: add}\cdot w \ R10, \&x \]
\[ \text{dec}\cdot w \ R10 \]
\[ \text{jz continue} \]
\[ \text{if Z bit is reset} \]
\[ \text{jmp repeat} \]
\[ \text{if Z bit is set} \]

\[ \text{continue: loop: jmp loop} \]
main

Initialize SP

Kim the Dog

R10 = 10
x = 0

repeat:

add.w R10, &x

dec.w R10

jz continue

continue:

Infinite Loop

R10
x

10 0 + 10
9 10 + 9
8 10 + 9 + 8
7 6 5 4 3 5 5
2
1 10 + 9 + ... + 1
0 exit

xc = 55
R10 = 0
Program "Conditional 1"

; Main loop here

mov.w #10, R10

repeat:
add.w R10, &x
dec.w R10

jz continue
jmp repeat

continue:

loop: jmp loop
Using TST instruction with conditional (jmp) statements

\[ \text{TST}.w \text{ dst} \quad (\text{TST}.b \text{ dst}) \]

**Status Bits**
- **N**: Set if destination is negative, reset if positive
- **Z**: Set if destination contains zero, reset otherwise
- **C**: Set
- **V**: Reset

**Example**

\[ \text{TST}.w \ R10 \quad jz \ mylabel \]

Jump to mylabel if Z bit is set (R10 is 0)
Otherwise continue from here
mov.w *N_delay, R10

repeat:

dec.w R10

jz continue

yes

continue:

no

jmp repeat

Operations: 
dec.w R10: 1 clock cycle
jz continue: 2 clock cycles
jmp repeat: 2 clock cycles

are repeated (N_delay - 1) number of times

Operations: 
dec.w R10: 1 clock cycle
jz continue: 2 clock cycles

are repeated once

assume N_delay = 0xFFFF

Total time wasted = \( \left(3 + \frac{5(N_{delay} - 1)}{327,670}\right)T_{\text{clock}}\)

= 327673 ms \(\approx 0.3\) sec delay
Total time wasted = \((3 + 5(N\_delay - 1))T\_clock\)

Determine \(N\_delay\) if a 0.25 sec delay is desired

ignoring (small)

\[3 + 5(N\_delay - 1)\mu s = 0.25\]

\[N\_delay = 50,000\]

Produce a delay of 0.5 sec

\[
\text{max } N\_delay = 0\timesFFFFF \text{ which gives 0.3 sec delay.}
\]

Solution: put two delays of 0.25 sec one after another.
program "Delay!"

N_delay: .set 50000
  
  mov.w #N_delay, R10
  
repeat:
  dec.w R10
  jz continue
  jmp repeat

continue:

loop: jmp loop

N_delay = 50,000

0.25 sec delay
program "Delay 2"

N_delay: .set 50000

mov.w #N_delay, R10

repeat1:
  dec.w R10
  jz continue1
  jmp repeat1

continue1:

mov.w #N_delay, R10

repeat2:
  dec.w R10
  jz continue2
  jmp repeat2

continue2:

loop: jmp loop

\[ N_{\text{delay}} = 50,000 \]