III. STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING TECHNIQUES

Control Structure: Loops

Control Structure: Loops

- While-Do Loop
  - while (TRUE) execute loop
  - loop body may not be executed even once

- Do-While Loop or Repeat-Until Loop
  - **do** loop **while** (TRUE) ➔ C
  - **repeat** loop **until** (TRUE) ➔ Pascal
  - loop body will be executed at least once

While-Do Loop

```
while (SI < BX) do {
    CX = CX * DI;
    SI++;
}
```

- checking of Boolean expression value is done at the start

```
while_begin:
    cmp SI, BX
    jnl   while_end
    mov AX, CX
    mul DI
    mov CX, AX
    inc   SI
    jmp while_begin
while_end:
```

Do-While Loop

```
do {
    CX = CX * DI;
    SI++;
} while (SI < BX);
```

- checking of Boolean expression value is done at the end

```
do_begin:
    mov AX, CX
    mul DI
    mov CX, AX
    inc  SI
    cmp SI, BX
    jl do_begin
do_end:
```

For Loop

- short-cut for **while-do** loop
- initialization;
  while(boolean_expr) do {
  loop_body;
  loop_variant;
  }

- for (initialization;
  boolean_expr;
  loop_variant) {
  loop_body;
  }
- For loop translates to a similar structure as a while-do loop

For Loop

```
x = 10;
while (x < 100) do {
    p = p * 3;
    x++;
}

for (x=10; x<100; x++)
    p = p * 3;
```

Loop Instruction

- A shortcut for several instructions
- Allows for a loop to execute several iterations (specified by ECX)

- **loop** label
  - **dec** ECX
  - **cmp** ECX, 0
  - **jne** label

Loop Instruction

for (ECX = x; ECX > 0; ECX--)

loop_body;

- **ECX** decrements by 1 at each iteration.

for ECX = x **downto** 1 **do**

loop_body;

Loop Instruction

```
for (ECX= 100; ECX > 0; ECX--)
    BX = BX + CX;
```

```
mov ECX, 100
begin_for:
    cmp ECX, 0
    jng end_for
    add BX, CX
    dec ECX
    jmp begin_for
end_for:
```

Loop Instruction

```
for (ECX= 100; ECX > 0; ECX--)
    BX = BX + CX;
```

```
mov ECX, 100
loop_begin:
    add BX, CX
    loop loop_begin
```