Control Instructions

- Controls the flow of the program
- Based on “events” e.g. calculation led to 0
- Uses flags to determine decision
- Branching
  - Unconditional – `JMP`
    - compare it with the GOTO statement in C

  - Conditional - `Jxx`
    - JZ (Jump on zero)
    - JNZ (Jump on not Zero)
    - JA
    - JAE
    - JC
    - JNC
  - Uses Flags.

CODE:

```
global _start

section .text
_start:

    jmp Begin
    ; will jump to Begin

NeverExecute:
    ; cause of previous jump    this will never execute

    mov rax, 0x10
    xor rbx, rbx

Begin:
    mov rax, 0x5

PrintHW:
    push rax ; To maintain Rax Original value Otherwise infinte loop will
happen

    ; print on screen

    mov rax, 1
    mov rdi, 1
    mov rsi, message
    mov rdx, mlen
    syscall

    pop rax ; to pop rax previous(original value)
    dec rax ; Rax --
    jnz PrintHW ; Jump if not zero


    ; exit gracefully

    mov rax, 60
    mov rdi, 11
    syscall




section .data

    message: db "Hello World! ", 0x0a
    mlen     equ $-message
```