Gambaran umum fungsi program

Program ini mengendalikan sebuah motor stepper melalui port A dari **8255 PPI**. Port B diset sebagai input dan digunakan untuk membaca tombol (push button) pada **bit0 (PB0)**. Tombol bersifat **active-LOW** (logika `0` = ditekan). Saat tombol ditekan program akan mematikan keluaran ke motor (mengeluarkan `0` di PORTA) dan menunggu sampai tombol dilepas (dengan debounce). Selama tombol tidak ditekan program berputar menjalankan 4 pola output yang menggerakkan stepper (langkah 1..4).

Definisi alamat port

```
PORTA    EQU 00H
PORTB    EQU 02H
PORTC    EQU 04H
PORT_CON EQU 06H
```

- `EQU` menamai alamat I/O perangkat 8255. (Asumsi: perangkat 8255 di-mapped ke alamat 00h,02h,04h,06h — ini tergantung hardware/rangkaian.)
- `PORT_CON` adalah alamat register kontrol 8255.

Segmen kode & origin

```
CODE SEGMENT
    ORG 100H
```

- Format .COM style / program sederhana (start di offset 100h). Bisa juga MASM/TASM dengan pengaturan yang sesuai.

Setting 8255 (Control Word)

```
MOV DX, PORT_CON
MOV AL, 10000010B
OUT DX, AL
```

- `DX` ← alamat register kontrol.
- `AL` ← `1000 0010b` (0x82).
- Ini adalah **control word** Mode Set untuk 8255:
    - D7 = 1 → mode set (menentukan konfigurasi).
    - D6-D5 = 00 → Group A mode = Mode 0 (input/output biasa).
    - D4 = 0 → Port A = **output**.
    - D3 = 0 → Port C (upper) = **output**.
    - D2 = 0 → Group B mode = Mode 0.
    - D1 = 1 → Port B = **input**.
    - D0 = 0 → Port C (lower) = **output**.
- Jadi: Port A output (untuk motor), Port B input (tombol), Port C output (tidak dipakai di program ini), Mode 0 (sederhana).

Loop utama (START)

```
START:
    ; sebelum setiap langkah cek PB0
    CALL CHECK_PB0

    MOV DX, PORTA
    ; langkah 1
    MOV AL, 00001001B
    OUT DX, AL
    CALL DELAY_SHORT

    CALL CHECK_PB0

    ; langkah 2
    MOV AL, 00000011B
    OUT DX, AL
    CALL DELAY_SHORT
```

```
CALL CHECK_PB0

; langkah 3
MOV AL, 00000110B
OUT DX, AL
CALL DELAY_SHORT

CALL CHECK_PB0

; langkah 4
MOV AL, 00001100B
OUT DX, AL
CALL DELAY_SHORT

JMP START
```

- `CALL CHECK_PB0` dipanggil **sebelum** setiap pola langkah dan juga **setelah** setiap langkah (dilihat: dipanggil sebanyak dua kali di tiap langkah—satu sebelum, satu setelah). Ini membuat program dapat merespons penekanan tombol dengan cepat (cek sering).
- `MOV DX, PORTA` meng-set alamat port A di DX (dipakai oleh `OUT DX, AL`).
- Pola yang dikirim ke `PORTA` berturut-turut: `00001001b`, `00000011b`, `00000110b`, `00001100b`.
    - Ini adalah pola 4-step untuk menggerakkan motor stepper (umumnya pola half-step/standard tergantung wiring motor). Masing-masing bit mewakili coil/coils dalam driver.
- `CALL DELAY_SHORT` memberi jeda antar langkah agar motor punya waktu bergerak.

Alur saat tombol ditekan: ketika `CHECK_PB0` mendeteksi tombol ditekan (active-LOW), ia memanggil `STOP_LOOP` yang akan mematikan PORTA (`0`) dan menunggu sampai tombol dilepas. Setelah dilepas, eksekusi kembali ke titik setelah `CALL CHECK_PB0` dan loop dilanjutkan.

Subrutin DELAY_SHORT

`DELAY_SHORT:`

```
    PUSH CX
    MOV CX, 04FFFh
LPS:
    LOOP LPS
    POP CX
    RET
```

- Delay sederhana memakai register CX dan instruksi `LOOP` (LOOP menurunkan CX dan lompat jika tidak nol).
- `PUSH CX` / `POP CX` untuk menjaga nilai CX caller.
- Nilai awal `04FFFh` (~0x4FFF = 20479 decimal) — lama delay bergantung pada clock CPU dan jumlah siklus tiap instruksi. Jadi **tidak absolut**; jika ingin delay tertentu (ms), harus kalibrasi terhadap clock atau gunakan timer hardware.
- `LOOP` sendiri memerlukan beberapa siklus CPU per iterasi, jadi total waktu = iterasi × siklus_per_loop / CPU_clock. Jangan asumsikan nilai waktu tanpa mengetahui CPU speed.

Subrutin CHECK_PB0 (debounce + branching)

```
CHECK_PB0:
    PUSH AX
    PUSH DX

    MOV DX, PORTB
    IN  AL, DX          ; baca port B
    ; tes bit0
    TEST AL, 00000001B
    JNZ  PB0_OK         ; jika bit0 = 1 -> tidak ditekan, lanjut
    ; jika bit0 = 0 -> kemungkinan ditekan, debounce: cek lagi
setelah cepat delay
    CALL DEBOUNCE_SHORT
    IN  AL, DX
    TEST AL, 00000001B
    JNZ  PB0_OK         ; jika sekarang 1, dianggap noise -> lanjut

    ; confirmed pressed -> stop motor sampai tombol dilepas
    CALL STOP_LOOP
```

```
PB0_OK:
    POP DX
    POP AX
    RET
```

- `PUSH AX` dan `PUSH DX` menyimpan register yang dipakai agar tidak merusak nilai caller.
- `IN AL, DX` membaca isi Port B (nilai input).
- `TEST AL, 00000001B` memeriksa **bit0** (PB0). `TEST` melakukan AND logis tanpa mengubah AL, dan `JNZ` (jump if not zero) akan terjadi kalau hasil `TEST` ≠ 0 ⇒ bit0 = 1.
    - Karena tombol **active-LOW**, **bit0 = 0** berarti tombol ditekan.
- Jika bit0=0, lakukan debounce: panggil `DEBOUNCE_SHORT` (delay kecil), baca lagi, jika masih 0 maka dianggap valid (bukan noise) → panggil `STOP_LOOP`.
- Setelah `STOP_LOOP` selesai (artinya tombol sudah dilepas), `CHECK_PB0` akan `RET` kembali ke pemanggil.
- Akhirnya `POP DX` / `POP AX` mengembalikan register.

Subrutin DEBOUNCE_SHORT

```
DEBOUNCE_SHORT:
    PUSH CX
    MOV CX, 0FFFh
DBL:
    LOOP DBL
    POP CX
    RET
```

- Delay singkat untuk debounce. Sama mekanisme seperti `DELAY_SHORT` tapi lebih kecil (`0FFFh`).

- Tujuan: menunggu sedikit waktu untuk memastikan sinyal stabil (mengurangi “bouncing”).

Subrutin STOP_LOOP

```
STOP_LOOP:
    ; set PORTA = 0 (matikan motor outputs)
    MOV DX, PORTA
    MOV AL, 00000000B
    OUT DX, AL

WAIT_RELEASE:
    MOV DX, PORTB
    IN  AL, DX
    TEST AL, 00000001B
    JZ  WAIT_RELEASE   ; masih ditekan -> tunggu
    ; debounced release: cek lagi cepat
    CALL DEBOUNCE_SHORT
    IN  AL, DX
    TEST AL, 00000001B
    JZ  WAIT_RELEASE

    RET
```

- Pertama mematikan output motor (`OUT PORTA, 0`) untuk memastikan motor tidak diberi sinyal saat tombol ditekan.
- Loop `WAIT_RELEASE` membaca PB0 dan menunggu sampai bit0 jadi `1` (dilepas).
- Setelah terdeteksi dilepas, lakukan debounce lagi untuk memastikan bukan noise sebelum keluar dari loop.
- Setelah dilepas, `STOP_LOOP RET` ke caller (`CHECK_PB0`), yang kemudian mengembalikan kontrol ke loop utama.