BỘ THÔNG TIN VÀ TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

****

BÁO CÁO BÀI TẬP LỚN
MÔN KĨ THUẬT VI XỬ LÝ

Môn học: Kĩ thuật vi xử lý
Giảng viên hướng dẫn: Phạm Hoàng Duy
Thực hiện: Nhóm 11 – Nhóm lớp 10
- Nguyễn Ngọc Long – B18DCCN353
- Đặng Tiến Đạo – B18DCCN133
- Tống Duy Khải – B18DCCN298
- Nguyễn Thị Ngọc Huyền – B18DCCN276

Hà Nội – 2020

MỤC LỤC

I. BÀI 1 – ĐIỀU KHIỂN ĐÈN GIAO THÔNG 3
1. Giải thích các bước thực hiện chương trình 3
2. Lưu đồ chương trình 6
3. Mã nguồn chương trình 7
II. BÀI 2 – ĐIỀU KHIỂN BẾP THÔNG MINH 12
1. Giải thích các bước thực hiện chương trình 12
2. Lưu đồ chương trình 17
3. Mã nguồn chương trình 18
III. BÀI 3 – MOTOR BƯỚC 23
1. Giải thích các bước thực hiện chương trình 23
2. Lưu đồ chương trình 25
3. Mã nguồn chương trình 26

I. BÀI 1 – ĐIỀU KHIỂN ĐÈN GIAO THÔNG

1. Giải thích các bước thực hiện chương trình

- Thiết bị ảo giả lập hệ thống đèn giao thông (Traffic Lights) sử dụng cổng số 4 (cổng 16 bit) để gửi thông tin điều khiển
- Sử dụng 12 bit từ bit 0 tới 11 cho 4 cụm đèn
    - Mỗi cụm đèn gồm 3 đèn Green, Yellow, Red (GYR – Xanh Vàng Đỏ)
    - Gửi bit 0 để tắt đèn, bit 1 để bật đèn
- 4 bit từ 12 đến 15 không được sử dụng nên đặt là 0
- Cách điều khiển đèn giao thông
    - Gửi từ điều khiển (2 bytes) ra cổng số 4
    - Các bit của từ điều khiển được đặt sao cho phù hợp với ý đồ điều khiển đèn theo quy ước về các bit ra ở trên

      <u>Ví dụ:</u>

      0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0
      G Y R G Y R G Y R G Y R
    - Dùng hàm 86h của ngắt 15h để tạo thời gian đợi nhằm giữ trạng thái vừa thiết lập của cụm đèn. Số micro giây được đưa vào thanh ghi BX:DX trước khi gọi ngắt.
- Giải thích chương trình

- Khởi tạo các bit của từ điều khiển để sau ta đưa ra cổng điều khiển số 4

```
.Data
; GYR GYR GYR GYR
R01 DW 0000_0010_0000_1000b   |
R03 DW 0000_0000_0100_0001b
R05 DW 0000_0000_0000_0000b
R1 DW 0000_0011_0000_1100b
R2 DW 0000_0010_1000_1010b
R3 DW 0000_1000_0110_0001b
R4 DW 0000_0100_0101_0001b
R5 DW 0000_0100_1001_0010b
; FEDC_BA9 876 543 210
all_red equ 0000_0010_0100_1001b
PORT EQU 4 ; output port
```

- Khởi tạo các hằng thời gian chờ

```
;wait 3 seconds (3 million microseconds)
WAIT_3_SEC_BX EQU 2Dh
WAIT_3_SEC_DX EQU 0C6C0h

;wait 32 seconds (32 million microseconds)
WAIT_32_SEC_BX EQU 1E8h
WAIT_32_SEC_DX EQU 4800h

;wait 1/3 seconds (1/3 million microseconds)
WAIT_13_SEC_BX EQU 4h
WAIT_13_SEC_DX EQU 93E0h
```

- Gọi hàm 2ch của ngắt 21h để lấy thời gian thực của hệ thống đồng thời so sánh điều kiện thời gian để chọn chu trình đèn.
    - Từ 23h30 đến 6h30 hôm sau: Chu trình giờ thấp điểm
    - Từ 6h30 đến 23h30: Chu trình đèn bình thường

```
Start:
    ; lay thoi gian tu he thong
    mov ah, 2ch    ; Chuc nang so 2Ch: doc thoi gian
    int 21h        ; Goi ngat

    cmp ch, 06
    jl toi
    je sosanh1
    cmp ch, 23
    jg toi
    je sosanh2
    jmp sang
    sosanh2:
        cmp cl,30
        jg toi
    jmp sang
    sosanh1:
        cmp cl,30
        jl toi
```

- 2 chế độ đèn trong chu trình đèn bình thường
    - Chế độ sáng:
        - Xanh sáng trong 32 giây: đưa thông tin bít đến cổng số 16 để điều khiển đèn và gọi hàm chờ 32s

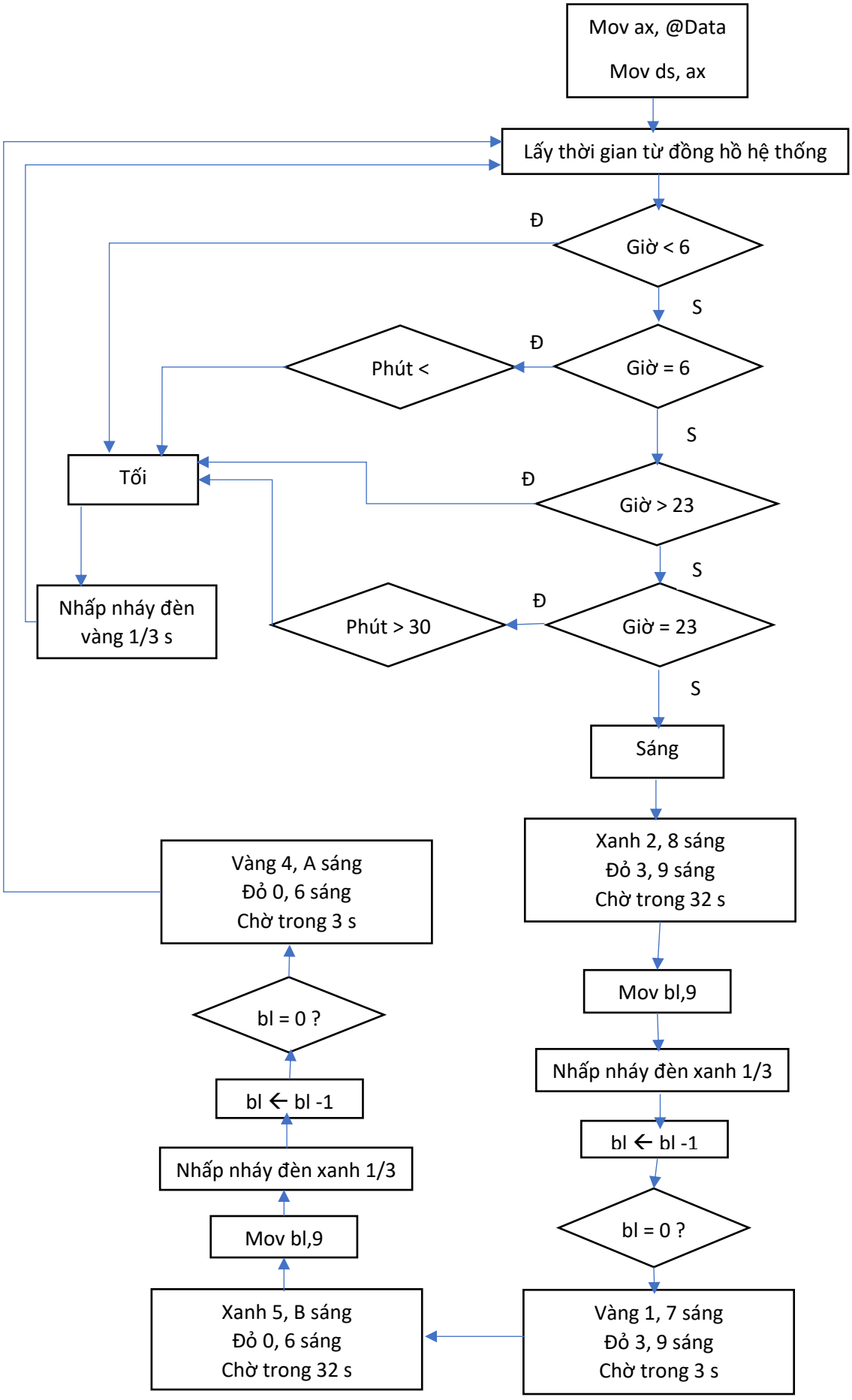
- 3 giây cuối đèn xanh sáng sẽ được nhấp nháy: Tạo một vòng lặp với CX = 9 nhấp nháy đèn và chờ trong 1/3s
- Đèn vàng sáng trong 3s: đưa thông tin bít đến cổng số 16 để điều khiển đèn và gọi hàm chờ 3s

```
Sang:
    lea si, R1
    mov ax, [si]
    out PORT, ax

    waitMacro WAIT_32_SEC_BX, WAIT_32_SEC_DX
    ; nhap nhay xanh trong 3s cuoi
    mov bl,9
        Lap:

            lea si, R01
            mov ax, [si]
            out PORT, ax

            lea si, R1
            mov ax, [si]
            out PORT, ax

            waitMacro WAIT_13_SEC_BX, WAIT_13_SEC_DX

            sub bl,1
            cmp bl,0
            jnz lap

    lea si, R2
    mov ax, [si]
    out PORT, ax

    waitMacro WAIT_3_SEC_BX, WAIT_3_SEC_DX

    lea si, R3
    mov ax, [si]
    out PORT, ax
    waitMacro WAIT_32_SEC_BX, WAIT_32_SEC_DX
    ; nhap nhay xanh trong 3s cuoi tan so
    mov bl,9
        Lap1:

            lea si, R03
            mov ax, [si]
            out PORT, ax

            lea si, R3
            mov ax, [si]
            out PORT, ax

            waitMacro WAIT_13_SEC_BX, WAIT_13_SEC_DX

            sub bl,1
            cmp bl,0
            jnz lap1

    lea si, R4
    mov ax, [si]
    out PORT, ax

    waitMacro WAIT_3_SEC_BX, WAIT_3_SEC_DX
    jmp Start
```

- Chế độ tối: Đưa thông tin bít đến cổng số 16 để điều khiển nhấp nháy đèn vàng với tần số 3Hz

```
Toi:
    lea si, R5
    mov ax, [si]
    out PORT, ax

    lea si, R05
    mov ax, [si]
    out PORT, ax

    waitMacro WAIT_13_SEC_BX, WAIT_13_SEC_DX
```

2. Lưu đồ chương trình



3. Mã nguồn chương trình

```
#start=Traffic_Lights.exe#

name "traffic"

.Model small
.Stack 100H
.Data
; GYR GYR GYR GYR
R01 DW 0000_0010_0000_1000b
R03 DW 0000_0000_0100_0001b
R05 DW 0000_0000_0000_0000b
R1 DW 0000_0011_0000_1100b
R2 DW 0000_0010_1000_1010b
R3 DW 0000_1000_0110_0001b
R4 DW 0000_0100_0101_0001b
R5 DW 0000_0100_1001_0010b
; FEDC_BA9 876 543 210
all_red equ 0000_0010_0100_1001b
PORT EQU 4 ; output port

;wait 3 seconds (3 million microseconds)
WAIT_3_SEC_BX EQU 2Dh
WAIT_3_SEC_DX EQU 0C6C0h

;wait 32 seconds (32 million microseconds)
WAIT_32_SEC_BX EQU 1E8h
WAIT_32_SEC_DX EQU 4800h

;wait 1/3 seconds (1/3 million microseconds)
WAIT_13_SEC_BX EQU 4h
WAIT_13_SEC_DX EQU 93E0h
```

```
.code
; define a macro
waitMacro macro t1, t2
mov cx, t1
mov dx, t2
mov ah, 86h
int 15h
waitMacro endm

main proc
; initilize the ds and es registers
mov ax, @Data
mov ds,ax

; set lights to Red for all direction
mov ax, all_red
out PORT, ax
waitMacro WAIT_3_SEC_BX, WAIT_3_SEC_DX

Start:
    ; lay thoi gian tu he thong
    mov ah, 2ch    ; Chuc nang so 2Ch: doc thoi gian
    int 21h        ; Goi ngat

    cmp ch, 06
    jl toi
    je sosanh1
    cmp ch, 23
    jg toi
    je sosanh2
    jmp sang
    sosanh2:
        cmp cl,30
```

```
    jg toi
jmp sang
sosanh1:
    cmp cl,30
    jl toi

Sang:
    lea si, R1
    mov ax, [si]
    out PORT, ax

    waitMacro WAIT_32_SEC_BX, WAIT_32_SEC_DX
    ; nhap nhay xanh trong 3s cuoi
    mov bl,9
        Lap:

            lea si, R01
            mov ax, [si]
            out PORT, ax

            lea si, R1
            mov ax, [si]
            out PORT, ax

            waitMacro WAIT_13_SEC_BX, WAIT_13_SEC_DX

            sub bl,1
            cmp bl,0
            jnz lap

    lea si, R2
    mov ax, [si]
    out PORT, ax
```

```
    waitMacro WAIT_3_SEC_BX, WAIT_3_SEC_DX

    lea si, R3
    mov ax, [si]
    out PORT, ax
    waitMacro WAIT_32_SEC_BX, WAIT_32_SEC_DX
    ; nhap nhay xanh trong 3s cuoi tan so
    mov bl,9
        Lap1:

            lea si, R03
            mov ax, [si]
            out PORT, ax

            lea si, R3
            mov ax, [si]
            out PORT, ax

            waitMacro WAIT_13_SEC_BX, WAIT_13_SEC_DX

            sub bl,1
            cmp bl,0
            jnz lap1

    lea si, R4
    mov ax, [si]
    out PORT, ax

    waitMacro WAIT_3_SEC_BX, WAIT_3_SEC_DX
    jmp Start

Toi:
```

```
        lea si, R5
        mov ax, [si]
        out PORT, ax

        lea si, R05
        mov ax, [si]
        out PORT, ax

        waitMacro WAIT_13_SEC_BX, WAIT_13_SEC_DX

    jmp Start
; end program
mov ah, 4CH
int 21H
main endp
end main
```

II. BÀI 2 – ĐIỀU KHIỂN BẾP THÔNG MINH

1. Giải thích các bước thực hiện chương trình

- Bước 1: Khởi tạo các biến dữ liệu
    - Các biến lưu giá trị giờ phút để đếm thời gian đun bếp (Dạng giờ:phút:giây, vừa để điều khiển bếp, vừa để hiển thị led): hour = 0, minute = 0, second = 0
    - Tạo biến heaterMode để lưu lại chế độ bếp đang đun (Các chế độ: 0 - Đun 100 độ, 1 - Giữ ấm 80 độ, 2 - Giữ ấm 60 độ). Khởi tạo đặt bằng 0
    - Nhiệt độ bếp khi bật chương trình ta tạm khởi tạo bằng 0
    - Hằng thời gian cho delay macro dùng để tạm dừng chương trình trong khoảng thời gian xác định (Trong chương trình này ta delay 3 giây)

```
hour dw 0h ; Luu lai thoi gian dun theo gio
minute dw 0h ; Luu lai thoi gian dun theo phut
second dw 0h ; Luu lai thoi gian dun theo giay
heaterMode db 0 ; Luu lai che do dun: 0 = Dun soi 100 do

heaterTempWhenProgStart db 0 ; Nhiet do cua bep khi bat

; Hang thoi gian cho delay macro (1s)
WAIT_3_SEC_CX EQU 2Dh
WAIT_3_SEC_DX EQU 0C6C0h
```

    - Macro delay chương trình

```
; Macro delay chuong trinh
delayMacro macro t1, t2
    mov cx, t1
    mov dx, t2
    mov ah, 86h
    int 15h
delayMacro endm
```

- Bước 2: Kiểm tra bếp bật hay chưa bằng cách lấy nhiệt độ đun hiện tại từ cổng 125, sau đó delay 3s và lấy nhiệt độ lần 2. Nếu nhiệt độ đo được lần

2 nhỏ hơn hoặc bằng nhiệt độ lần 1 thì bếp đang tắt, ta bật bếp.

```
; Kiem tra bep da bat hay chua (Phan nang cao)
checkHeater:
    in al, 125
    mov heaterTempWhenProgStart, al
    delayMacro WAIT_3_SEC_CX, WAIT_3_SEC_DX
    mov ah, 2ch
    int 21h
    in al, 125
    cmp al, heaterTempWhenProgStart
    jle on
```

- Bước 3: Ghi nhận và kiểm tra thời gian đã chạy chương trình, kiểm tra nhiệt độ để lựa chọn chế độ đun

    - Gọi chương trình con timer mỗi đầu nhãn lệnh đun bếp để ghi nhận lại thời gian đã chạy chương trình và hiển thị led đếm một cách liên tục. Chương trình con này sử dụng hàm 2ch của ngắt 21h để lấy thời gian hệ thống.

```
; Bo dem thoi gian
timer proc
    mov al, dh
    mov ah, 2ch
    int 21h

    call showLed

    cmp dh, al
    jg incSec
    jmp ok

    incSec:
    ;tang=giay+1
    ;kiem tra neu giay=60 thi tang phut
        add second, 1h
        cmp second, 3ch ; 3ch (hex) = 60 (dec)
        je incMin
        jmp ok

    incMin:
    ;reset giay=0 -> phut=phut+1;
    ;kiem tra neu phut=60 thi tang gio;
        mov second, 0h ; reset giay ve 0
        add minute, 1h
        cmp minute, 3ch ; 3ch (hex) = 60 (dec)
        je incHour
        jmp ok

    incHour:
    ;reset phut=0, gio=gio+1;
        mov minute, 0h ; reset phut ve 0
        add hour, 1h

    ok:
        ret
timer endp


; Hien thi den led
showLed proc
    push ax
    push dx
    mov si, 64h ; 64h = 100 (dec)
    mov ax, hour ; chuyen gio vao ax de nhan 100
    mul si
    add ax, minute
    out 199, ax
    pop dx
    pop ax
    ret
showLed endp
```

- Kiểm tra thời gian để lựa chọn chế độ đun
    - Ở nhãn lệnh bolingAndKeep100: hour < 1 đúng thì đun 100 độ, sai thì nhảy sang nhãn lệnh keepWarm80 để đun 80 độ

```
bolingAndKeep100:
    call timer
    mov [heaterMode], 0 ; Gan che do dun = 0: Dun soi den 100 do
    cmp hour, 1h
    jge keepWarm80

    in al, 125
    cmp al, 100
    jle on
    jg off
    jmp bolingAndKeep100
```

    - Ở nhãn lệnh keepWarm80: hour < 3 đúng thì đun 100 độ, sai thì nhảy sang nhãn lệnh keepWarm60 để giữ ấm 60 độ

```
keepWarm80:
    call timer
    mov [heaterMode], 1 ; Gan che do dun = 1: Giu am 80 do
    cmp hour, 3h
    jge keepWarm60

    in al, 125
    cmp al, 80
    jle on
    jg off
    jmp keepWarm80


keepWarm60:
    call timer
    mov [heaterMode], 2 ; Gan che do dun = 2: Giu am 60 do
    in al, 125
    cmp al, 60
    jle on
    jg off
    jmp keepWarm60
```

- Kiểm tra nhiệt độ:

  Ở đầu mỗi nhãn lệnh ta cũng ghi nhận lại chế độ đun (heaterMode) theo quy ước của người lập trình để nhãn lệnh bật/tắt bếp biết bếp đang ở chế độ đun nào, từ đó giúp quay lại lệnh đun một cách chính xác.

  Quy ước: 0 - Đun 100 độ, 1 - Giữ ấm 80 độ, 2 - Giữ ấm 60 độ

  - Nhiệt độ bếp > Nhiệt độ cần đun của chế độ đun hiện tại: Tắt bếp

```
; Tat bep dun bang cach gui bit 1 ra cong 127
; So sanh che do bep dun de nhay ve nhan lenh thich hop
off:
    mov al, 0
    out 127, al

    cmp heaterMode, 0
    jge bolingAndKeep100

    cmp heaterMode, 1
    jge keepWarm80

    cmp heaterMode, 2
    jge keepWarm60
```

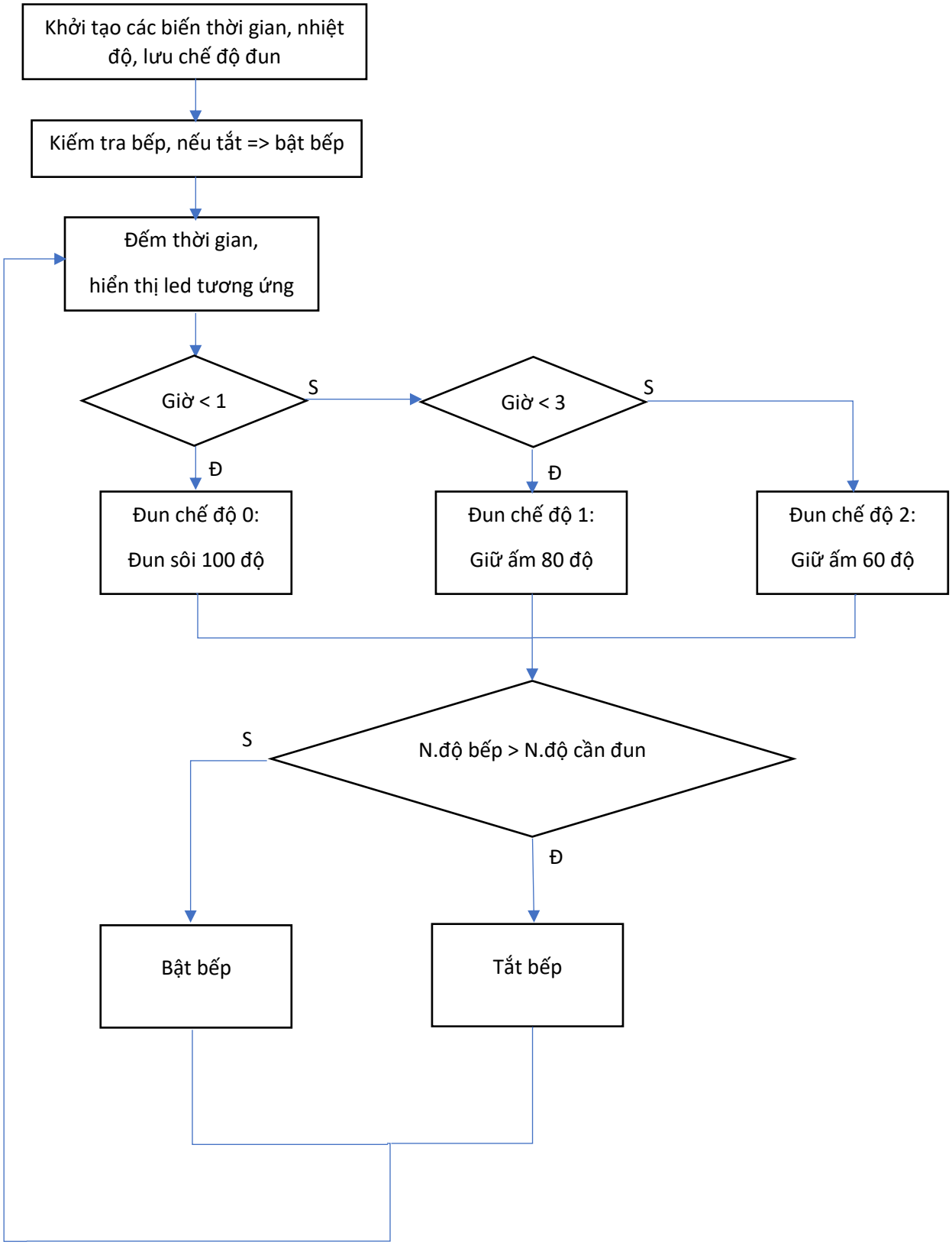
  - Nhiệt độ bếp ≤ Nhiệt độ cần đun của chế độ đun hiện tại: Bật bếp

```
; Bat bep dun bang cach gui bit 1 ra cong 127
; So sanh che do bep dun de nhay ve nhan lenh thich hop
on:
    mov al, 1
    out 127, al

    cmp heaterMode, 0
    jge bolingAndKeep100

    cmp heaterMode, 1
    jge keepWarm80

    cmp heaterMode, 2
    jge keepWarm60
```

- Bước 4: Quay lại bước 3

  Ở đầu mỗi nhãn lệnh tương ứng với 1 chế độ đun ta gọi chương trình con timer. Nhờ việc so sánh điều kiện nhiệt độ và nhảy lệnh mà chúng ta đã tạo nên một vòng lặp vô hạn, do đó thời gian đun bếp luôn được ghi nhận và việc kiểm soát nhiệt độ cũng luôn được thực hiện liên tục.

2. Lưu đồ chương trình



3. Mã nguồn chương trình

```
#start=thermometer.exe#
#start=led_display.exe#

#make_bin#

name "bep_dun_nuoc_ml"

.model small
.data
    hour dw 0h ; Luu lai thoi gian dun theo gio
    minute dw 0h ; Luu lai thoi gian dun theo phut
    second dw 0h ; Luu lai thoi gian dun theo giay
    heaterMode db 0 ; Luu lai che do dun: 0 = Dun soi 100 do, 1 =
Giu am 80 do, 2 = Giu am 60 do

    heaterTempWhenProgStart db 0 ; Nhiet do cua bep khi bat

    ; Hang thoi gian cho delay macro (3s)
    WAIT_3_SEC_CX EQU 2Dh
    WAIT_3_SEC_DX EQU 0C6C0h
.code

; Macro delay chuong trinh
delayMacro macro t1, t2
    mov cx, t1
    mov dx, t2
    mov ah, 86h
    int 15h
delayMacro endm

main proc
    mov ax, cs
```

```
mov ds, ax

; Kiem tra bep da bat hay chua (Phan nang cao)
checkHeater:
    in al, 125
    mov heaterTempWhenProgStart, al
    delayMacro WAIT_3_SEC_CX, WAIT_3_SEC_DX
    mov ah, 2ch
    int 21h
    in al, 125
    cmp al, heaterTempWhenProgStart
    jle on

bolingAndKeep100:
    call timer
    mov [heaterMode], 0 ; Gan che do dun = 0: Dun soi den 100 do
    cmp hour, 1h
    jge keepWarm80

    in al, 125
    cmp al, 100
    jle on
    jg off
    jmp bolingAndKeep100

keepWarm80:
    call timer
    mov [heaterMode], 1 ; Gan che do dun = 1: Giu am 80 do
    cmp hour, 3h
    jge keepWarm60

    in al, 125
    cmp al, 80
```

```
    jle on
    jg off
    jmp keepWarm80

keepWarm60:
    call timer
    mov [heaterMode], 2 ; Gan che do dun = 2: Giu am 60 do
    in al, 125
    cmp al, 60
    jle on
    jg off
    jmp keepWarm60

; Bat bep dun bang cach gui bit 1 ra cong 127
; So sanh che do bep dun de nhay ve nhan lenh thich hop
on:
    mov al, 1
    out 127, al

    cmp heaterMode, 0
    jge bolingAndKeep100

    cmp heaterMode, 1
    jge keepWarm80

    cmp heaterMode, 2
    jge keepWarm60


; Tat bep dun bang cach gui bit 1 ra cong 127
; So sanh che do bep dun de nhay ve nhan lenh thich hop
off:
    mov al, 0
```

```
        out 127, al

        cmp heaterMode, 0
        jge bolingAndKeep100

        cmp heaterMode, 1
        jge keepWarm80

        cmp heaterMode, 2
        jge keepWarm60

main endp

; Bo dem thoi gian
timer proc
    mov al, dh
    mov ah, 2ch
    int 21h

    call showLed

    cmp dh, al
    jg incSec
    jmp ok

    incSec:
        add second, 1h
        cmp second, 3ch ; 3ch (hex) = 60 (dec)
        je incMin
        jmp ok

    incMin:
        mov second, 0h ; reset giay ve 0
```

```
        add minute, 1h
        cmp minute, 3ch ; 3ch (hex) = 60 (dec)
        je incHour
        jmp ok

    incHour:
        mov minute, 0h ; reset phut ve 0
        add hour, 1h

    ok:
        ret
timer endp

; Hien thi den led
showLed proc
    push ax
    push dx
    mov si, 64h ; 64h = 100 (dec)
    mov ax, hour ; chuyen gio vao ax de nhan 100
    mul si
    add ax, minute
    out 199, ax
    pop dx
    pop ax
    ret
showLed endp
end main
```

III. BÀI 3 – MOTOR BƯỚC

1. Giải thích các bước thực hiện chương trình

- Đầu tiên, ta đặt 1 biến để đếm vòng quay. Sau đó chúng ta thiết lập chiều quay cho mô-tơ: 1 là chiều thuận (quay theo chiều kim đồng hồ từ phải sang trái), 2 là chiều nghịch (quay ngược chiều kim đồng hồ từ trái sang phải). Cài đặt các biến và in ra màn hình vị trí hiện tại của kim trỏ. Sau đó chúng ta thiết lập chiều để xác định chiều quay.

```
jmp start

; ========= data ===============
cout dw 0
; bin data for clock-wise
; half-step rotation:
datcw     db 0000_0110b
          db 0000_0100b
          db 0000_0011b
          db 0000_0010b

; bin data for counter-clock-wise
; half-step rotation:
datccw    db 0000_0011b
          db 0000_0001b
          db 0000_0110b
          db 0000_0010b

start:
mov si,0
mov cx,0
mov ax,0;
out 199,ax

set_datcw:
mov bx,offset datcw
jmp next_step
set_datccw:
mov bx,offset datccw
jmp next_step
```

- Hàm đợi: Kiểm tra bit thứ 7 của stepper motor. Sau đó chúng ta thực hiện lấy bước tiếp theo, đưa ra màn hình bước tiếp theo đó. Sau đó chúng ta kiểm tra đã hết 1 chu kì chưa. Nếu đã hết 1 chu kì thì quay về, nếu không thì nhảy đến next_step.

```
next_step:
; motor sets top bit when it's ready to accept new command
wait:     in al, 7
          test al, 10000000b
          jz wait

mov al, [bx][si]
out 7, al
inc si

cmp si, 4
jb next_step
mov si, 0
```

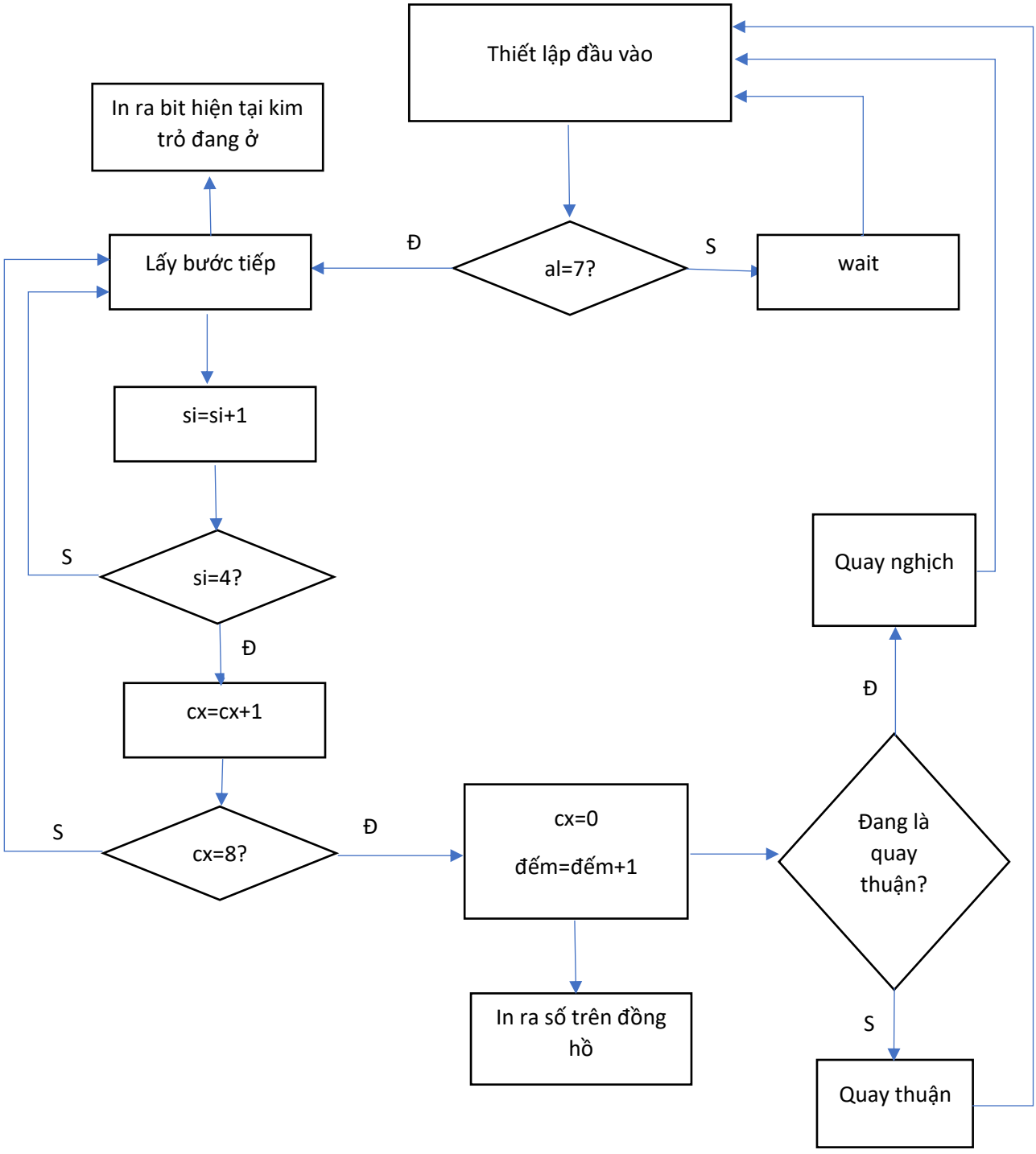
- Tăng số lần chạy trong chu kì lên. So sánh để biết được đã chạy xong 1 chu kì chưa, nếu chưa thì nhảy đến next_step. Cộng thêm 1 cho đếm, và in ra đồng hồ.

```
inc cx
cmp cx, 8
jb  next_step

mov cx,0
inc cout
mov ax, cout
out 199,ax
```

- Thiết lập vòng quay cho motor, so sánh vòng đang quay với quay thuận, nếu vòng đang quay là quay thuận thì nhảy sang quay nghịch, còn nếu không thì nhảy sang quay thuận.

```
cmp bx, offset datcw
je set_datccw
jne set_datcw
```

2. Lưu đồ chương trình



3. Mã nguồn chương trình

```
#start=stepper_motor.exe#
#start=led_display.exe#
name "stepper"
#make_bin#
steps_before_direction_change = 8 ; 32 (decimal)

jmp start

; ========= data ===============
cout dw 0
; bin data for clock-wise
; half-step rotation:
datcw    db 0000_0110b
         db 0000_0100b
         db 0000_0011b
         db 0000_0010b

; bin data for counter-clock-wise
; half-step rotation:
datccw   db 0000_0011b
         db 0000_0001b
         db 0000_0110b
         db 0000_0010b

start:
mov si,0
mov cx,0
mov ax,0;
out 199,ax
set_datcw:
mov bx,offset datcw
jmp next_step
```

```
set_datccw:
mov bx,offset datccw
jmp next_step

next_step:
; motor sets top bit when it's ready to accept new command
wait:   in al, 7
        test al, 10000000b
        jz wait

mov al, [bx][si]
out 7, al
inc si

cmp si, 4
jb next_step
mov si, 0

inc cx
cmp cx, 8
jb  next_step

mov cx,0
inc cout
mov ax, cout
out 199,ax

cmp bx, offset datcw
je set_datccw
jne set_datcw
```