CSE2006
Microprocessor & Interfacing

Module – 5
Introduction to Peripheral Interfacing II

Dr. E. Konguvel
Assistant Professor (Sr. Gr. 1),
Dept. of Embedded Technology,
School of Electronics Engineering (SENSE),
konguvel.e@vit.ac.in
9597812810

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Module 5: Introduction Peripheral Interfacing II

- Serial Communication Interface – 8251
- Analog-to-Digital Converter Interfacing
- **Digital-to-Analog Converter Interfacing**
- Programmable Keyboard & Display Interface – 8279

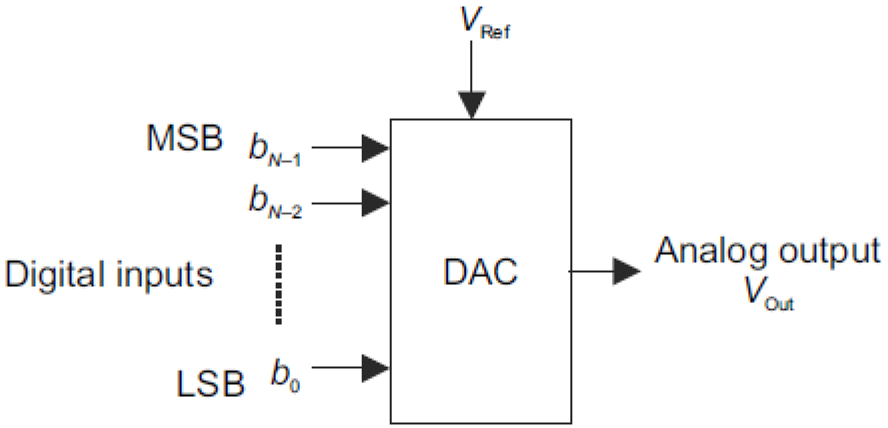
Digital to Analog Converters

- Introduction & Types
- Specifications for DAC ICs
- DAC AD7523
- DAC 0800
- Interfacing
- Problems

DAC – Introduction

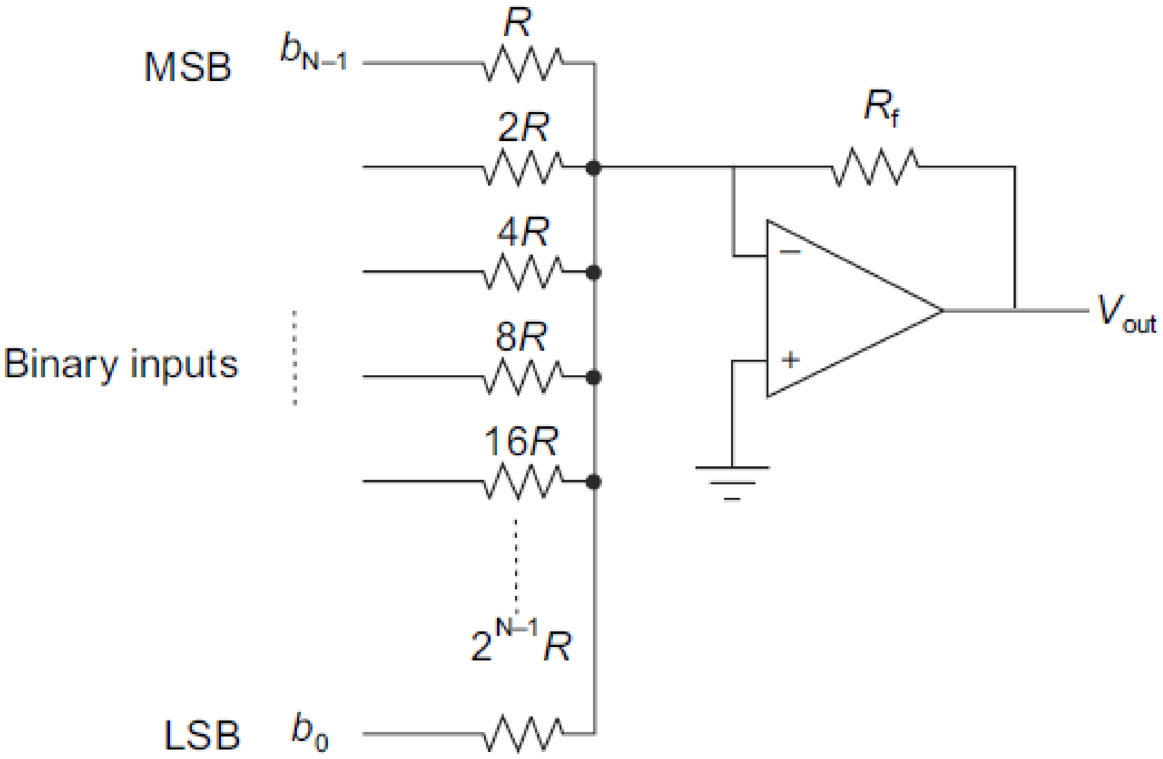
- Digital-to-Analog Converter (DAC) has the ability to convert digital signals to analog signals.
- DAC is a process in which digital words are applied to the input of the DAC and an analog output signal is generated to represent the respective digital input.
- In this conversion process, an ‘N’-bit digital data can be mapped into a single analog output voltage.
- So, analog output of the DAC is a voltage that is some fraction of a reference voltage.

$$V_{out} = K \times V_{Ref}$$



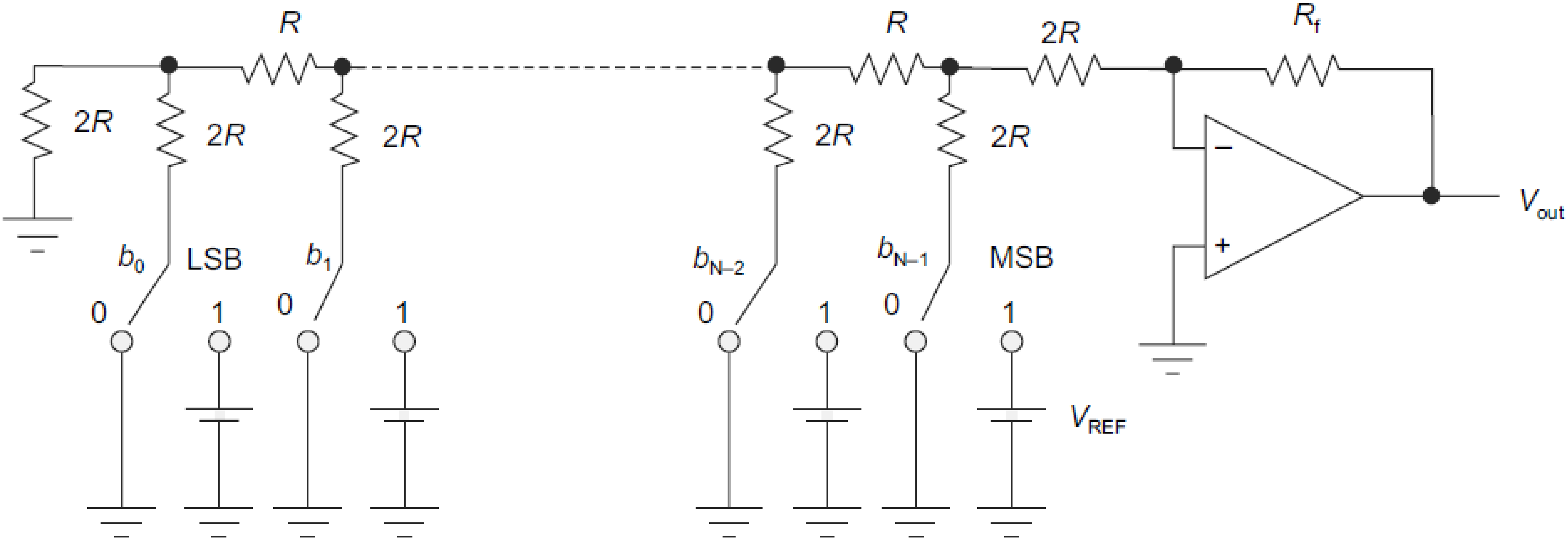
DAC – Introduction

- Classifications:
  - Weighted binary DAC
  - R-2R ladder
- N-bit Weighted binary DAC:



DAC – Introduction

- N-bit R-2R ladder DAC:

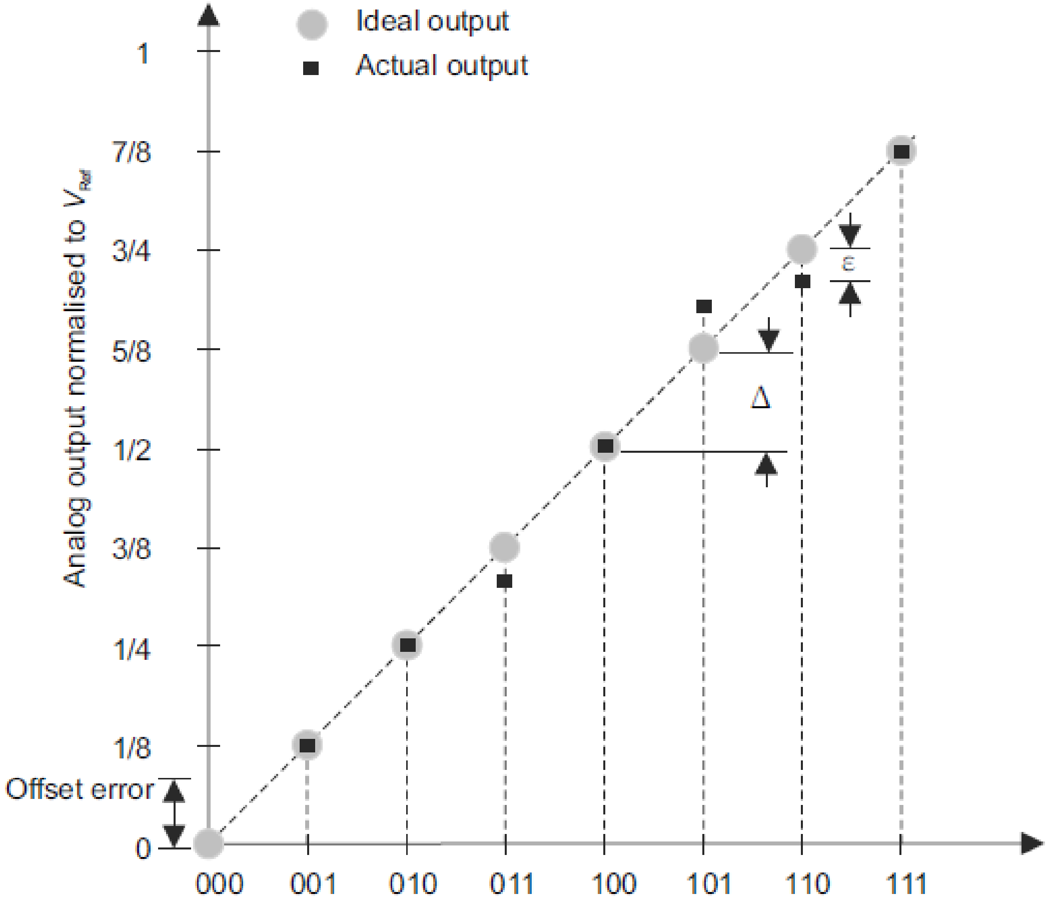


DAC – Specifications for DAC ICs

- **Resolution:**

$$\text{Resolution} = \frac{\text{Reference Voltage}}{\text{Number of Steps}} = \frac{V}{2^{N-1}}$$

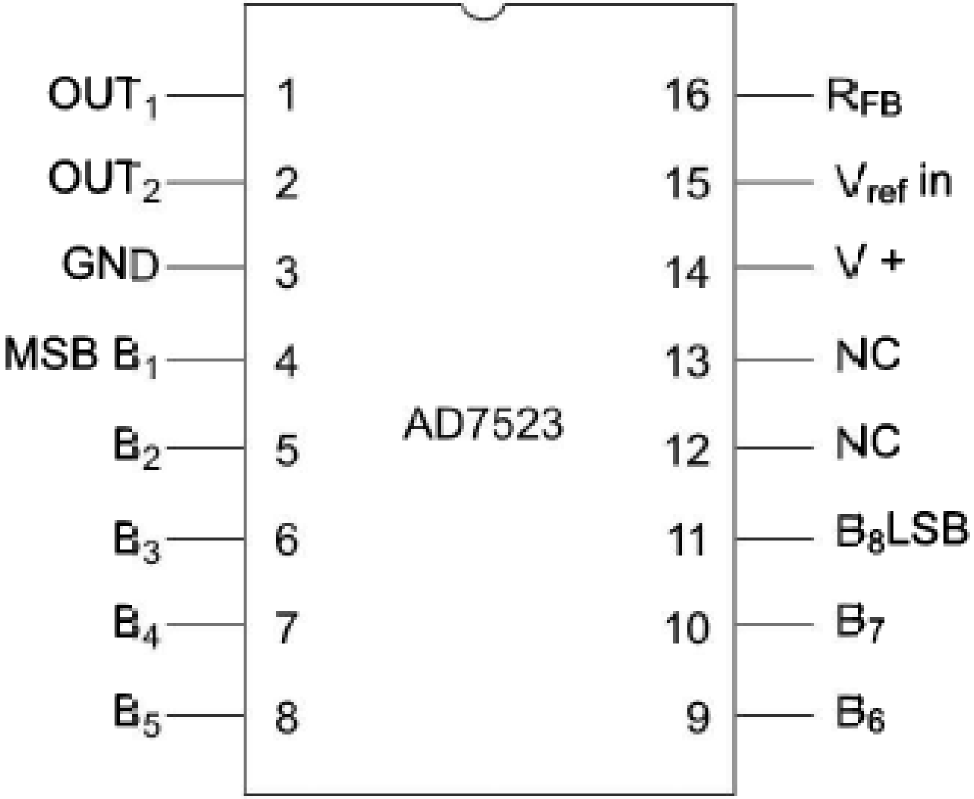
- **Accuracy:**
  - ±0.25%
- **Offset/Zero Scale Error:**
- **Linearity:**



- **Temperature Sensitivity:**
  - ± 50 ppm/°C.
- **Settling Time:**
  - Time taken to settle within half LSB, 500ns

DAC AD 7523

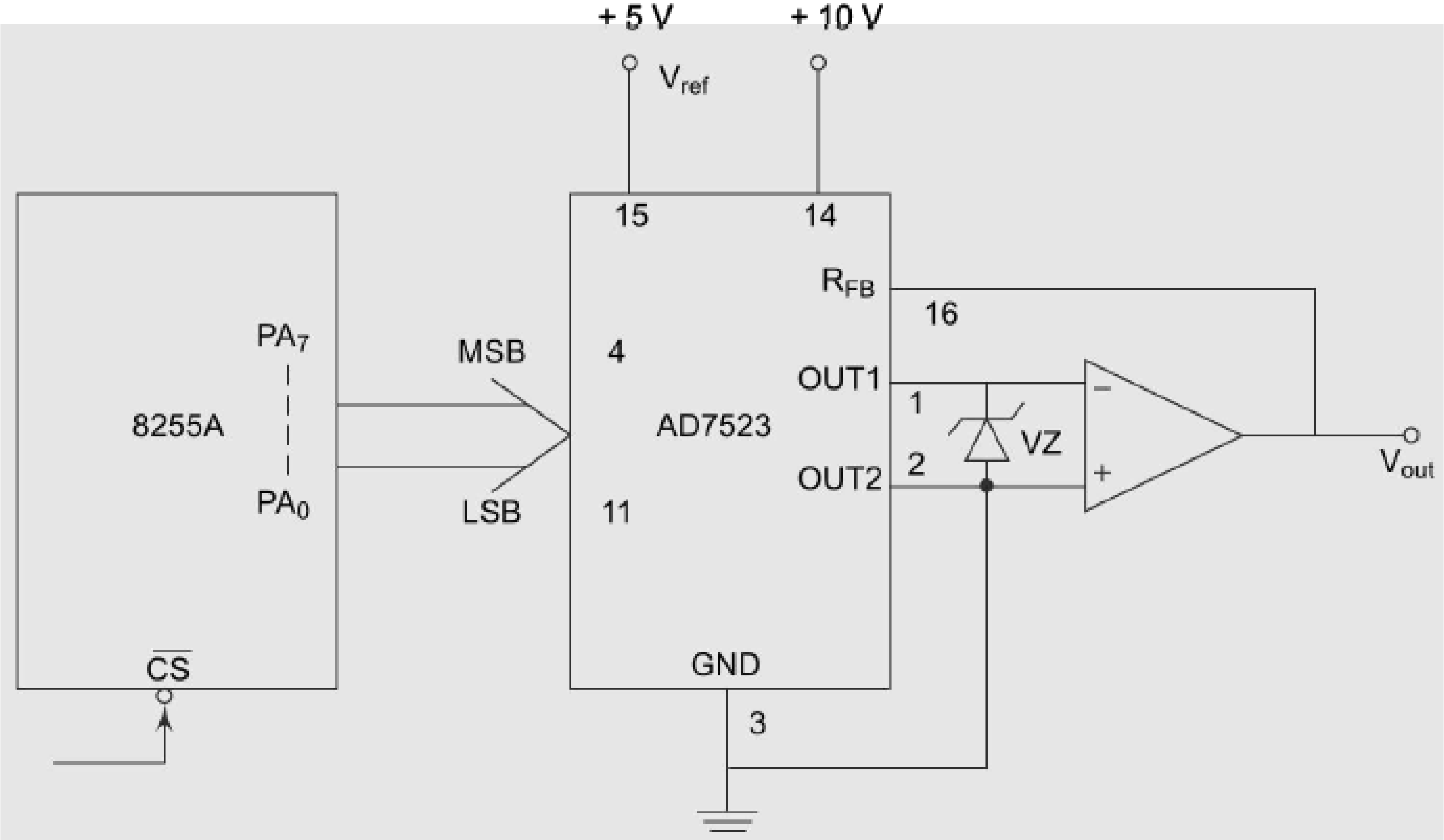
- R-2R (R = 10k)
- Supply range: 5V – 15V
- Max Analog output = +10V



DAC AD 7523

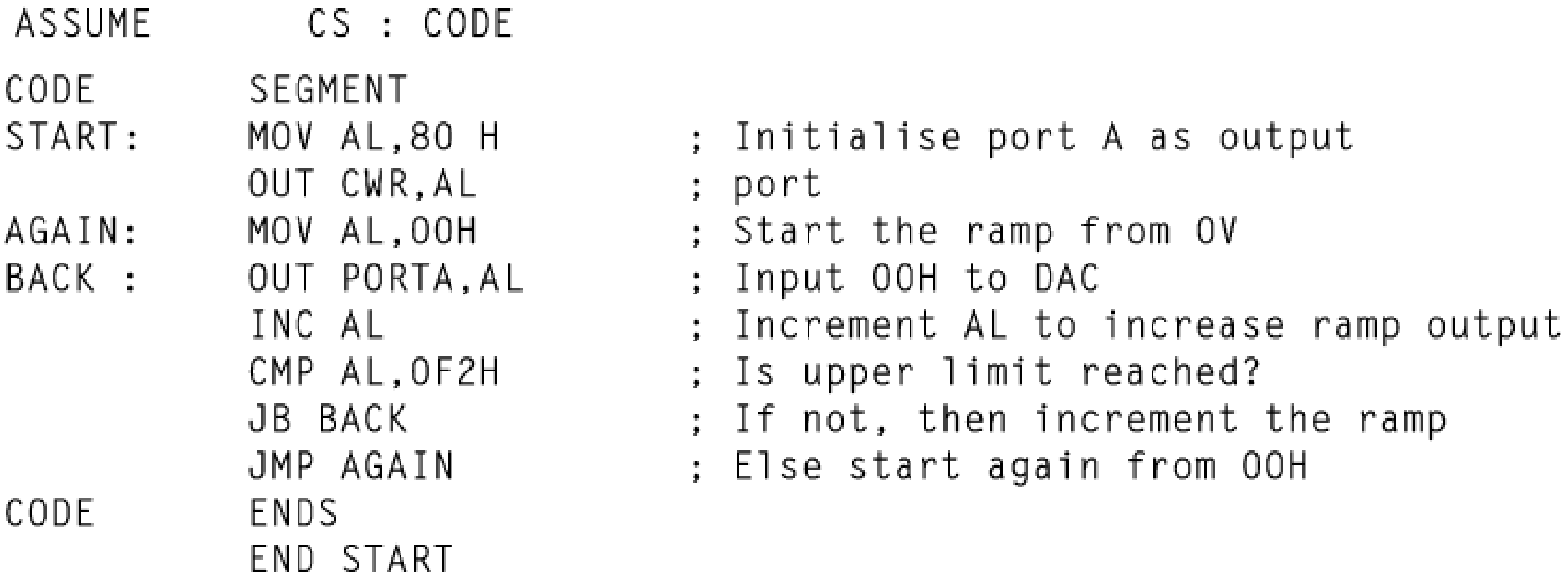
Problem 1

Interface DAC AD7523 with an 8086 CPU running at 8 MHz and write an assembly language program to generate a sawtooth waveform of period 1 ms with V_{max} 5V.



DAC AD 7523

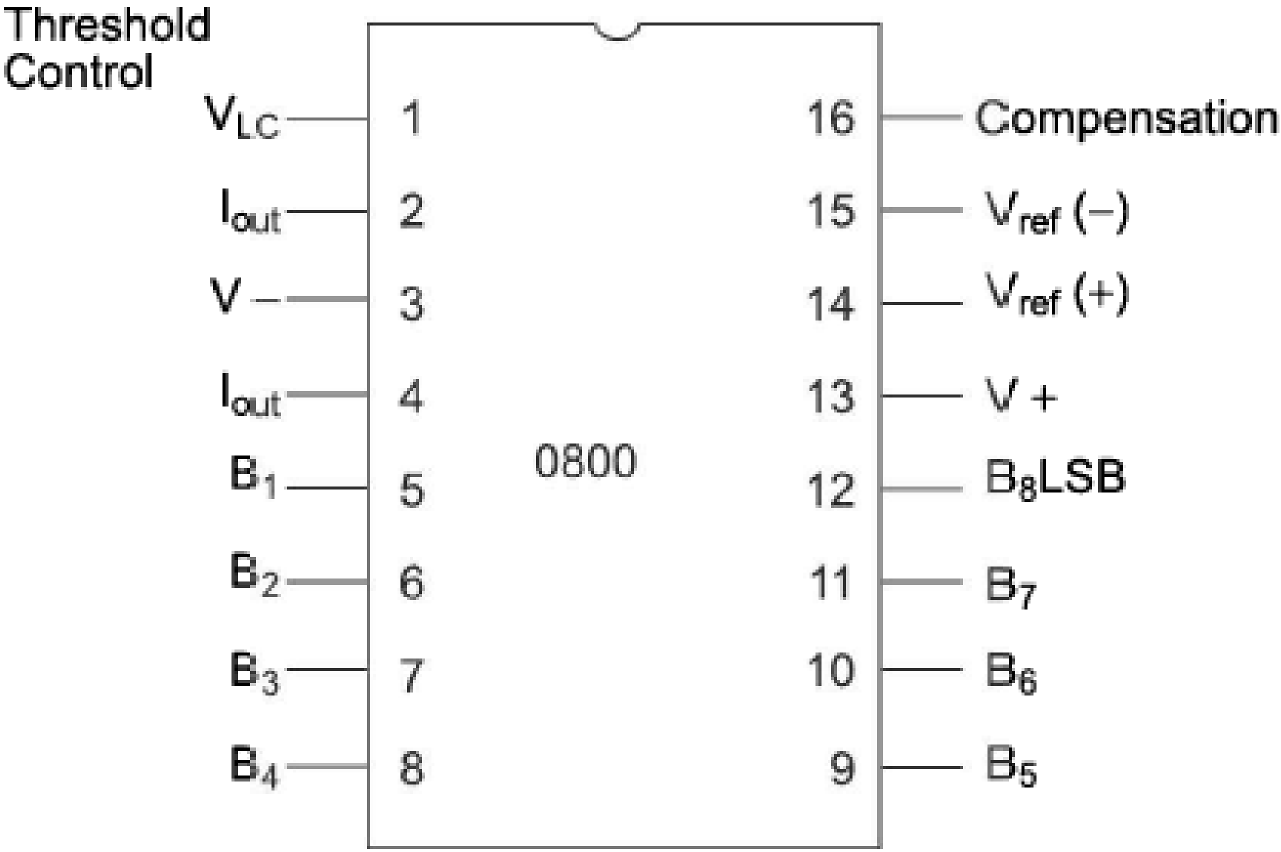
ALP Program



```
 ASSUME        CS : CODE
CODE        SEGMENT
START:      MOV AL,80 H        ; Initialise port A as output
            OUT CWR,AL         ; port
AGAIN:      MOV AL,00H         ; Start the ramp from 0V
BACK :      OUT PORTA,AL       ; Input 00H to DAC
            INC AL             ; Increment AL to increase ramp output
            CMP AL,0F2H        ; Is upper limit reached?
            JB BACK            ; If not, then increment the ramp
            JMP AGAIN          ; Else start again from 00H
CODE        ENDS
            END START
```

DAC 0800

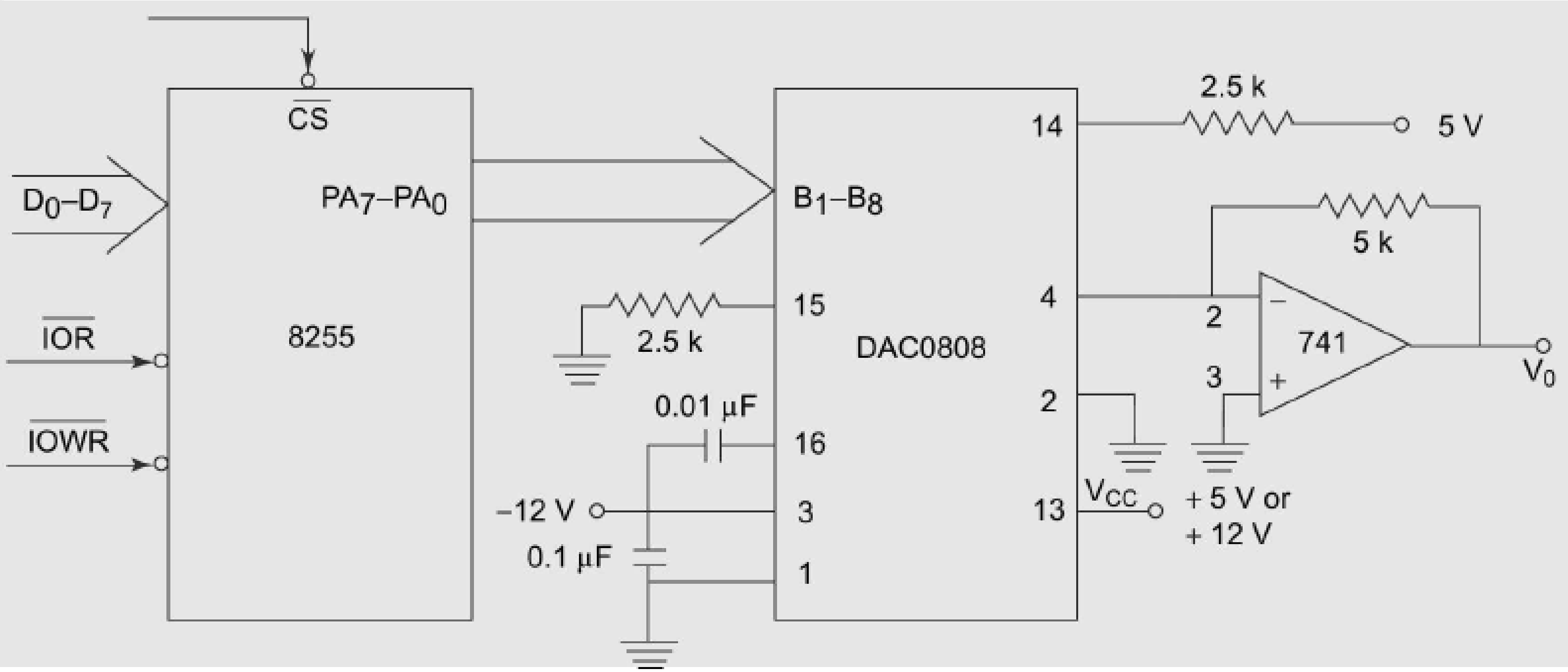
- Settling time 100ms
- Operating range +4.5V to +18V



DAC 0800

Problem 2

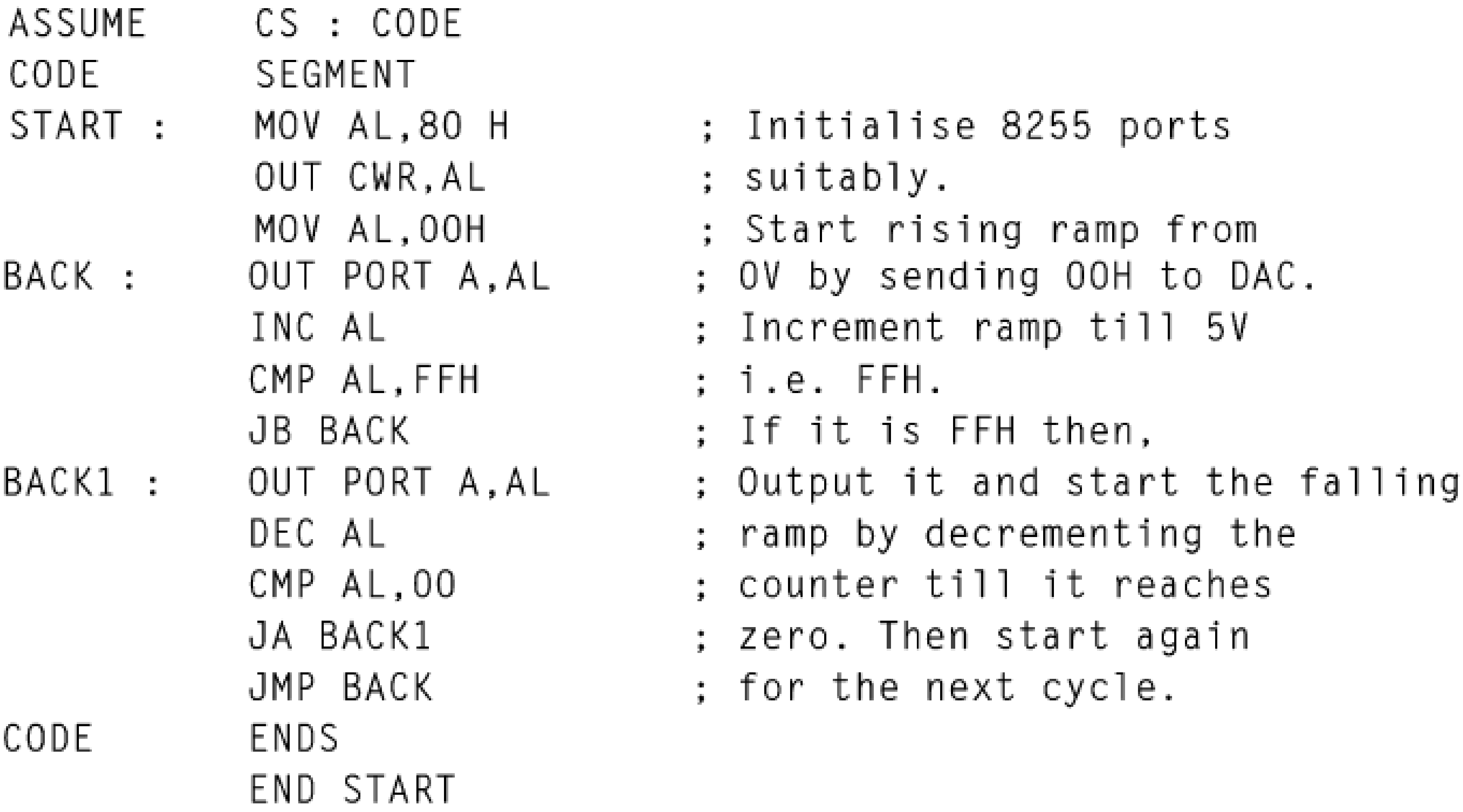
Write an assembly language program to generate a triangular wave of frequency 500 Hz using the interfacing circuit given. The 8086 system operates at 8 MHz. The amplitude of the triangular wave should be +5 V.



The V_{ref+} should be tied to +5 V to generate a wave of +5V amplitude. The required frequency of the output is 500 Hz, i.e. the period is 2 ms. Assuming the wave to be generated is symmetric, the waveform will rise for 1 ms and fall for 1 ms. This will be repeated continuously. In the previous program, we have already written an instruction sequence for period 1 ms. Using the same instruction sequence one can derive this triangular waveform.

DAC 0800

ALP Program



```
ASSUME     CS : CODE
CODE       SEGMENT
START :    MOV AL,80 H        ; Initialise 8255 ports
           OUT CWR,AL         ; suitably.
           MOV AL,00H         ; Start rising ramp from
BACK :     OUT PORT A,AL      ; 0V by sending 00H to DAC.
           INC AL             ; Increment ramp till 5V
           CMP AL,FFH         ; i.e. FFH.
           JB BACK            ; If it is FFH then,
BACK1 :    OUT PORT A,AL      ; Output it and start the falling
           DEC AL             ; ramp by decrementing the
           CMP AL,00          ; counter till it reaches
           JA BACK1           ; zero. Then start again
           JMP BACK           ; for the next cycle.
CODE       ENDS
           END START
```