Segment Definition

- The CPU has several segment registers:
  - CS (code segment).
  - SS (stack segment).
  - DS (data segment).
  - ES (extra segment).
  - FS, GS (supplemental segments available on 386s, 486s and Pentiums).

- Every instruction and directive must correspond to a segment.

- Normally a program consists of three segments: the stack, the data, and the code segments.
Segment Definition

- Model definition.
- .MODEL SMALL
  - Most widely used memory model.
  - The code must fit in 64k.
  - The data must fit in 64k.
- .MODEL MEDIUM
  - The code can exceed 64k.
  - The data must fit in 64k.
- .MODEL COMPACT
  - The code must fit in 64k.
  - The data can exceed 64k.
- MEDIUM and COMPACT are opposites.
Segment Definition

- **.MODEL LARGE**
  - Both code and data can exceed 64k.
  - No single set of data can exceed 64k.

- **.MODEL HUGE**
  - Both code and data can exceed 64k.
  - A single set of data can exceed 64k.

- **.MODEL TINY**
  - Used with COM files.
  - Both code and data must fit in a single 64k segment.
Segment Definition

- Segment definition formats:
  - Simplified segment definition.
  - Full segment definition.

- The Simplified segment definition uses the following directives to define the segments:
  - .STACK
  - .DATA
  - .CODE
  - These directives mark the beginning of the segments they represent.
Segment Definition

- The full segment definition uses the following directives to define the segments:
  - Label SEGMENT [options]
  - ;Statements belonging to the segment.
  - Label ENDS
  - The label must follow naming conventions previously discussed.
Segment Definition

;SIMPLIFIED SEGMENT DEFINITION

.MODEL SMALL
.STACK 64

.CODE
BEGIN PROC FAR
MOV AX,@DATA
MOV DS,AX
MOV AX,N1
ADD AX,N2
MOV SUM,AX
MOV AX,4CH
INT 21H
BEGIN ENDP

DATA
N1 DW 1432H
N2 DW 4365H
SUM DW 0H

BEGIN PROC FAR
ASSUME CS:CDSEG,DS:DTSEG,SS:STSEG
MOV AX,DTSEG
MOV DS,AX
MOV AX,N1
ADD AX,N2
MOV AX,4CH
INT 21H
BEGIN ENDP

CDSEG SEGMENT
END BEGMIN

DATA
N1 DW 1432H
N2 DW 4365H
SUM DW 0H

STSEG SEGMENT
DB 64 DUP(?)
STSEG ENDS

DTSEG SEGMENT
N1 DW 1432H
N2 DW 4365H
SUM DW 0H
DTSEG ENDS

STSEG ENDS

STSEG ENDS
Addressing Modes

These are the different ways in which data may be accessed by the microprocessor.

- Immediate.
- Register.
- Memory.
  - Direct.
  - Register indirect.
  - Register relative.
  - Based indexed.
  - Relative based indexed.
Immediate

- Directly accessible to the EU.
- The address is part of the instruction.
- Useful in initializations.
- MOV EAX,1111000B
- MOV CL, 0F1H
Register

- Directly accessible to the EU.
- Most compact and fastest executing instructions.
- Operands are encoded in the instruction.
- MOV EBX,EDX
- MOV AL,CL
Memory

- When reading or writing to memory the execution unit passes an offset value, the effective address, to the bus interface unit which then computes the physical address.

- Memory.
  - Direct.
  - Register indirect.
  - Register relative.
  - Based indexed.
  - Relative based indexed.
Direct

EA = \{\text{operand}\}
PA = \{DS\} \times 16 + \{\text{operand}\}

- Simplest memory addressing mode.
- Access to simple variables.
- MOV EAX,DS:SUM
- MOV CL,DS:COUNT+5
- MOV DS:[500H],EDX
Register Indirect

\[ \text{EA} = \begin{cases} (\text{EBX}) \\ (\text{EDI}) \\ (\text{ESI}) \end{cases} \]

\[ \text{PA} = \{\text{DS}\} \times 16 + \begin{cases} (\text{EBX}) \\ (\text{EDI}) \\ (\text{ESI}) \end{cases} \]

- MOV EAX, DS:[EBX]
- MOV DS:[EDI], EDX
Register Relative

\[
EA = \left\{ \begin{array}{c}
(EBX) \\
(EBP) \\
(EDI) \\
(ESI)
\end{array} \right\} + \left\{ \begin{array}{c}
8 \text{ bit displacement} \\
16 \text{ bit displacement}
\end{array} \right\}
\]

\[
PA = \left\{ \begin{array}{c}
DS \\
SS \\
DS \\
DS
\end{array} \right\} \times 16 + \left\{ \begin{array}{c}
(EBX) \\
(EBP) \\
(EDI) \\
(ESI)
\end{array} \right\} + \left\{ \begin{array}{c}
8 \text{ bit displacement} \\
16 \text{ bit displacement}
\end{array} \right\}
\]

- Access to one dimensional arrays.
- MOV EAX,DS:ARRAY[EBX]
- MOV DS:MESSAGE[EDI], DL
Relative Based Indexed

\[
EA = \left\{ \frac{(EBX)}{(EBP)} \right\} + \left\{ \frac{(EDI)}{(ESI)} \right\} + \left\{ 8 \text{ bit displacement} \right\} + \left\{ 16 \text{ bit displacement} \right\}
\]

\[
PA = \left\{ \frac{DS}{SS} \right\} \times 16 + \left\{ \frac{(EBX)}{(EBP)} \right\} + \left\{ \frac{(EDI)}{(ESI)} \right\} + \left\{ 8 \text{ bit displacement} \right\} + \left\{ 16 \text{ bit displacement} \right\}
\]

- Used to access two dimensional arrays or arrays contained in structures.
- MOV DS:ARRAY[EBX][EDI],EAX
Accessing Arrays

- One dimensional arrays.
  - MOV DS:ARRAY[ESI*SF],EDX
  - SF = Scaling factor for data size.

- Two dimensional arrays.
  - MOV DS:ARRAY[EBX*SF*SR][ESI*SF],EDX
  - SF = Scaling factor for data size.
  - SR = Size of row.
Accessing Arrays

Assume the following array definition:

```
ARRAY DD 00112233H, 44556677H, 88990011H
```

Begin:
```
LEA EBX,DS:ARRAY
L1:  
    MOV EAX,DS:[EBX]
    INC EBX
    JMP L1
```

Begin:
```
MOV ESI,O
L1:  
    MOV EAX,DS:ARRAY[ESI]
    INC ESI
    JMP L1
```

Begin:
```
MOV ESI,O
L1:  
    MOV EAX,DS:ARRAY[ESI*4]
    INC ESI
    JMP L1
```
Alignment

- It is best to align words with even numbered addresses, and double words to addresses divisible by four, but this is not necessary.
- The alignment allows for more efficient memory access, but it is less flexible.
Immediate - Memory

- When reading or writing to memory using immediate addressing mode, the programmer must specify the data size otherwise the assembler will default to the largest possible data size that processor handles.

- Use the following directives:
  - Byte ptr.
  - Word ptr.
  - Dword ptr.

- MOV DS:BYTE PTR VAR,2H
Procedures

- Also known as subroutines, these sets of instructions usually perform a single task.
- They are reusable code, that can be executed as often as needed by calling it.
- Procedures save memory, but the calling of a procedure takes a small amount of time.
Procedures

- **Format**
  - Name PROC [NEAR or FAR]
  - Subroutine code
  - RET
  - ENDP

- Global procedures are defined as FAR.
- Local procedures are defined as NEAR.
Procedures

- **CALL destination**
  - Calls a subroutine at location destination.
  - Different addressing modes may be used for destination.
    - CALL DELAY
    - CALL EBX
    - CALL ARRAY[BX]

- **RET**
  - Returns execution of program to location stored in stack.
  - NEAR or FAR is dependent on procedure definition.