City Event Booking System - Full Execution Simulation

This document provides a complete line-by-line execution simulation of the City Event Booking System for a specific customer interaction. We'll trace through exactly how program execution flows, including all jumps and procedure calls.

Customer Scenario

- Customer selects Concert (C)
- Orders 3 tickets
- Adds Parking Pass and VIP Upgrade as extras
- Uses coupon code "EVT"
- Confirms booking

Program Execution Simulation

Program Start and Initialization

```assembly
MAIN PROC
    MOV AX, @DATA       ; AX = address of data segment
    MOV DS, AX          ; DS = data segment address
```

> Execution starts at MAIN, initializes DS register to access program data

```assembly
    CALL PrintNewline   ; Call procedure to print newline
```

> Execution jumps to PrintNewline procedure

assembly

```
PrintNewline PROC
    MOV tempDX, DX       ; Save DX to memory (DX = unknown initial value)
    MOV tempAX, AX       ; Save AX to memory (AX = data segment address)

    LEA DX, newline      ; DX = address of newline string
    MOV AH, 9            ; AH = 9 (DOS string output function)
    INT 21H              ; Call DOS interrupt to display newline

    MOV AX, tempAX       ; Restore AX (data segment address)
    MOV DX, tempDX       ; Restore DX (original value)
    RET                  ; Return to caller
PrintNewline ENDP
```

> Execution returns to MAIN after displaying newline

assembly

```
    LEA DX, welcomeMsg1  ; DX = address of first welcome message
    CALL PrintString     ; Call PrintString procedure
```

> Execution jumps to PrintString procedure

assembly

```
PrintString PROC
    MOV tempAX, AX       ; Save AX to memory

    MOV AH, 9            ; AH = 9 (DOS string output function)
    INT 21H              ; Call DOS interrupt to display string in DX

    MOV AX, tempAX       ; Restore AX
    RET                  ; Return to caller
PrintString ENDP
```

> Execution returns to MAIN after displaying welcome message 1

assembly

```
    CALL PrintNewline
    LEA DX, welcomeMsg2
    CALL PrintString
    CALL PrintNewline
    LEA DX, welcomeMsg3
    CALL PrintString
```

> Similar procedure calls display remaining welcome messages (details omitted for brevity)

Event Menu Display and Selection

assembly

```
    CALL PrintNewline
    LEA DX, eventCategoryMsg
    CALL PrintString
    CALL PrintNewline
    LEA DX, concertMsg
    CALL PrintString
    CALL PrintNewline
    LEA DX, sportsMsg
    CALL PrintString
    CALL PrintNewline
    LEA DX, workshopMsg
    CALL PrintString
    CALL PrintNewline
    LEA DX, categoryEnd
    CALL PrintString
```

> Multiple procedure calls display the event menu to the customer

assembly

```
EventSelection:
    CALL PrintNewline
    LEA DX, promptSelectEvent    ; "Please select an event category (C/S/W): "
    CALL PrintString

    CALL GetCharToUpper          ; Get user input with uppercase conversion
```

> Execution jumps to GetCharToUpper procedure

```assembly
GetCharToUpper PROC
    MOV tempBX, BX              ; Save BX to memory

    MOV AH, 1                   ; AH = 1 (DOS character input function)
    INT 21H                     ; Wait for keypress, result in AL
                                ; *** CUSTOMER ENTERS 'C' ***

    CMP AL, 'a'                 ; Check if lowercase (not in this case)
    JB NotLowerCase             ; Jump if below 'a'
    CMP AL, 'z'
    JA NotLowerCase
    SUB AL, 32                  ; Convert lowercase to uppercase (not needed here)

NotLowerCase:
    MOV BX, tempBX              ; Restore BX
    RET                         ; Return with character in AL ('C')
GetCharToUpper ENDP
```

> Execution returns to EventSelection with 'C' in AL

```assembly
    MOV eventType, AL           ; Save event type 'C' to memory

    CMP AL, 'C'                 ; Compare with 'C'
    JE ConcertSelected          ; Jump if equal (which it is)
    CMP AL, 'S'                 ; These comparisons are skipped due to jump
    JE SportSelected
    CMP AL, 'W'
    JE WorkshopSelected

    CALL PrintNewline           ; Error handling (skipped in this scenario)
    LEA DX, errorInvEvent
    CALL PrintString
    JMP EventSelection
```

> AL = 'C', so execution jumps to ConcertSelected

```assembly
ConcertSelected:
    MOV SI, 0                   ; SI = 0 (index for concert price)
    JMP GetEventPrice           ; Jump to price lookup
```

> Execution jumps to GetEventPrice

```assembly
GetEventPrice:
    MOV AX, eventPrices[SI]     ; AX = 500 (concert price at index 0)
    MOV eventPrice, AX          ; Save event price to memory
    JMP TicketQuantity          ; Jump to ticket quantity input
```

> Execution jumps to TicketQuantity

Ticket Quantity Input

```assembly
TicketQuantity:
    CALL PrintNewline
    LEA DX, promptNumTickets    ; "How many tickets do you want? (max 5): "
    CALL PrintString

    MOV AH, 1                   ; AH = 1 (DOS character input)
    INT 21H                     ; Get keypress into AL
                               ; *** CUSTOMER ENTERS '3' ***

    SUB AL, 30H                 ; Convert ASCII '3' to number 3
    MOV numTickets, AL          ; Save to memory

    CMP numTickets, 0           ; Validate input
    JLE InvalidTickets          ; Not taken (3 > 0)
    CMP numTickets, 5           ; Check maximum
    JG TooManyTickets           ; Not taken (3 ≤ 5)
    JMP ExtrasPrompt            ; Jump to extras section
```

> Execution jumps to ExtrasPrompt

Extras Selection

```
assembly
```

```
ExtrasPrompt:
    CALL PrintNewline

    LEA DX, extrasHeader        ; Display extras menu header
    CALL PrintString
    CALL PrintNewline
    LEA DX, parkingMsg          ; Display parking option
    CALL PrintString
    CALL PrintNewline
    LEA DX, merchandiseMsg      ; Display merchandise option
    CALL PrintString
    CALL PrintNewline
    LEA DX, vipMsg              ; Display VIP option
    CALL PrintString
    CALL PrintNewline
    LEA DX, categoryEnd
    CALL PrintString

    CALL PrintNewline
    LEA DX, promptExtras        ; "Would you like to add extras? (Y/N): "
    CALL PrintString

    CALL GetCharToUpper         ; Get user input
                                ; *** CUSTOMER ENTERS 'Y' ***

    CMP AL, 'Y'
    JE ExtrasInit               ; Jump if yes (which it is)
    CMP AL, 'N'                 ; Skip this check due to jump
    JE CalculateBill

    CALL PrintNewline           ; Error handling (skipped)
    LEA DX, errorInvInput
    CALL PrintString
    JMP ExtrasPrompt
```

> AL = 'Y', so execution jumps to ExtrasInit

```
assembly
```

```
ExtrasInit:
    MOV extrasCount, 0          ; Initialize extras counter to 0
```

Execution falls through to SelectExtras

```assembly
SelectExtras:
    CMP extrasCount, 3          ; Check if max extras reached
    JE CalculateBill            ; Not taken (count is 0)

    CALL PrintNewline
    LEA DX, promptWhichExtra    ; "Select extra (P/M/V) or X to finish: "
    CALL PrintString

    CALL GetCharToUpper         ; Get user input
                                ; *** CUSTOMER ENTERS 'P' (Parking) ***

    CMP AL, 'P'
    JE AddParking               ; Jump if P (which it is)
    CMP AL, 'M'                 ; Skip these checks due to jump
    JE AddMerchandise
    CMP AL, 'V'
    JE AddVIP
    CMP AL, 'X'
    JE CalculateBill

    CALL PrintNewline           ; Error handling (skipped)
    LEA DX, errorInvExtra
    CALL PrintString
    JMP SelectExtras
```

AL = 'P', so execution jumps to AddParking

```assembly
AddParking:
    CMP hasParking, 1           ; Check if already selected
    JNE SetParking              ; Jump if not (which it isn't)

    CALL PrintNewline           ; Already selected message (skipped)
    LEA DX, alreadySelectedMsg
    CALL PrintString
    JMP SelectExtras
```

hasParking = 0, so execution jumps to SetParking

```
assembly
```

```
SetParking:
    MOV hasParking, 1            ; Set flag to selected
    INC extrasCount              ; Increment counter to 1
    JMP SelectExtras             ; Return for more selection
```

> Execution jumps back to SelectExtras for second selection

```
assembly
```

```
SelectExtras:
    CMP extrasCount, 3           ; Check if max extras reached
    JE CalculateBill             ; Not taken (count is 1)

    CALL PrintNewline
    LEA DX, promptWhichExtra     ; "Select extra (P/M/V) or X to finish: "
    CALL PrintString

    CALL GetCharToUpper          ; Get user input
                                 ; *** CUSTOMER ENTERS 'V' (VIP) ***

    CMP AL, 'V'
    JE AddVIP                    ; Jump if V (which it is)
```

> AL = 'V', so execution jumps to AddVIP

```
assembly
```

```
AddVIP:
    CMP hasVIP, 1                ; Check if already selected
    JNE SetVIP                   ; Jump if not (which it isn't)

    CALL PrintNewline            ; Already selected message (skipped)
    LEA DX, alreadySelectedMsg
    CALL PrintString
    JMP SelectExtras
```

> hasVIP = 0, so execution jumps to SetVIP

```assembly
SetVIP:
    MOV hasVIP, 1               ; Set flag to selected
    INC extrasCount             ; Increment counter to 2
    JMP SelectExtras            ; Return for more selection
```

> Execution jumps back to SelectExtras for third selection

```assembly
SelectExtras:
    CMP extrasCount, 3          ; Check if max extras reached
    JE CalculateBill            ; Not taken (count is 2)

    CALL PrintNewline
    LEA DX, promptWhichExtra    ; "Select extra (P/M/V) or X to finish: "
    CALL PrintString

    CALL GetCharToUpper         ; Get user input
                                ; *** CUSTOMER ENTERS 'X' (Finish) ***

    CMP AL, 'X'
    JE CalculateBill            ; Jump if X (which it is)
```

> AL = 'X', so execution jumps to CalculateBill

Bill Calculation

```assembly
CalculateBill:
    MOV AX, eventPrice          ; AX = 500 (concert price)
    MOV BL, numTickets          ; BL = 3 (tickets)
    MOV BH, 0                   ; BH = 0 (clear high byte)
    MUL BX                      ; AX = 500 * 3 = 1500
    MOV subtotal, AX            ; subtotal = 1500
```

> Ticket cost calculated: 3 tickets × 500 = 1500

assembly

```
    CMP hasParking, 1          ; Check if parking selected
    JNE CheckMerchandise       ; Not taken (hasParking = 1)
    MOV SI, 0                  ; SI = 0 (index for parking price)
    MOV AX, extraPrices[SI]    ; AX = 100 (parking price)
    ADD subtotal, AX           ; subtotal += 100 = 1600
```

> Parking cost added: 1500 + 100 = 1600

assembly

```
CheckMerchandise:
    CMP hasMerch, 1            ; Check if merchandise selected
    JNE CheckVIP               ; Jump if not (which it isn't)
    MOV SI, 2                  ; This part is skipped
    MOV AX, extraPrices[SI]
    ADD subtotal, AX
```

> hasMerch = 0, so execution jumps to CheckVIP

assembly

```
CheckVIP:
    CMP hasVIP, 1              ; Check if VIP selected
    JNE CheckGroupDiscount     ; Not taken (hasVIP = 1)
    MOV SI, 4                  ; SI = 4 (index for VIP price)
    MOV AX, extraPrices[SI]    ; AX = 250 (VIP price)
    ADD subtotal, AX           ; subtotal += 250 = 1850
```

> VIP cost added: 1600 + 250 = 1850

```assembly
CheckGroupDiscount:
    MOV discount, 0             ; Initialize discount to 0

    CMP numTickets, 3           ; Check if 3+ tickets for group discount
    JL CouponPrompt             ; Not taken (numTickets = 3)

    MOV AX, subtotal            ; AX = 1850
    MOV BX, 15                  ; BX = 15 (15% discount)
    MUL BX                      ; AX = 1850 * 15 = 27750
    MOV BX, 100                 ; BX = 100
    DIV BX                      ; AX = 27750 / 100 = 277 (remainder 50 in DX)
    MOV discount, AX            ; discount = 277
```

> Group discount calculated: 15% of 1850 = 277

Coupon Processing

```assembly
CouponPrompt:
    CALL PrintNewline
    LEA DX, promptCoupon        ; "Do you have a coupon code? (Y/N): "
    CALL PrintString

    CALL GetCharToUpper         ; Get user input
                                ; *** CUSTOMER ENTERS 'Y' ***

    CMP AL, 'Y'
    JE EnterCoupon              ; Jump if Y (which it is)
    CMP AL, 'N'                 ; Skip this check due to jump
    JE NoCouponSelected

    CALL PrintNewline           ; Error handling (skipped)
    LEA DX, errorInvInput
    CALL PrintString
    JMP CouponPrompt
```

> AL = 'Y', so execution jumps to EnterCoupon

```assembly
EnterCoupon:
    CALL PrintNewline
    LEA DX, enterCouponCode      ; "Enter coupon code (3 characters): "
    CALL PrintString

    MOV CX, 3                    ; CX = 3 (counter for 3 characters)
    LEA SI, couponCode           ; SI points to coupon storage

ReadCouponLoop:
    CALL GetCharToUpper          ; Get character input
                                 ; *** CUSTOMER ENTERS 'E', then 'V', then 'T' ***
    MOV [SI], AL                 ; Store character at SI
    INC SI                       ; Move to next position
    LOOP ReadCouponLoop          ; Repeat until CX = 0
```

> After loop, couponCode contains "EVT"

```assembly
    MOV CX, 3                    ; Reset counter
    LEA SI, couponCode           ; SI points to entered code
    LEA DI, validCoupon          ; DI points to valid code "EVT"

ValidateCouponLoop:
    MOV AL, [SI]                 ; AL = entered character
    MOV BL, [DI]                 ; BL = valid character
    CMP AL, BL                   ; Compare them
    JNE InvalidCoupon            ; Not taken (they match)
    INC SI                       ; Next position in entered code
    INC DI                       ; Next position in valid code
    LOOP ValidateCouponLoop      ; Repeat until CX = 0
```

> Coupon validated successfully (matches "EVT")

```assembly
    MOV AX, subtotal          ; AX = 1850
    MOV BX, 10                ; BX = 10 (10% discount)
    MUL BX                    ; AX = 1850 * 10 = 18500
    MOV BX, 100               ; BX = 100
    DIV BX                    ; AX = 18500 / 100 = 185
    MOV couponDisc, AX        ; couponDisc = 185

    CALL PrintNewline
    LEA DX, validCouponMsg    ; "Coupon successfully applied! 10% discount."
    CALL PrintString
    JMP FinalizeTotal         ; Jump to finalize
```

> Coupon discount calculated: 10% of 1850 = 185 Execution jumps to FinalizeTotal

Finalize Totals and Display Bill

```assembly
FinalizeTotal:
    MOV AX, subtotal          ; AX = 1850
    SUB AX, discount          ; AX = 1850 - 277 = 1573
    SUB AX, couponDisc        ; AX = 1573 - 185 = 1388
    MOV totalBill, AX         ; totalBill = 1388

    CALL PrintNewline
    CALL PrintNewline
    LEA DX, billHeader        ; "YOUR BOOKING DETAILS"
    CALL PrintString

    CALL PrintNewline
    LEA DX, eventTypeMsg      ; "Event Type: "
    CALL PrintString

    MOV AL, eventType         ; AL = 'C'
    CMP AL, 'C'
    JE DisplayConcert         ; Jump if C (which it is)
    CMP AL, 'S'               ; Skip these checks due to jump
    JE DisplaySports
    CMP AL, 'W'
    JE DisplayWorkshop
```

> AL = 'C', so execution jumps to DisplayConcert

assembly

```
DisplayConcert:
    LEA DX, concertType         ; "Concert"
    JMP PrintEventType          ; Jump to print it
```

Execution jumps to PrintEventType

assembly

```
PrintEventType:
    CALL PrintString            ; Print the event type

    CALL PrintNewline
    LEA DX, numTicketsMsg       ; "Number of tickets: "
    CALL PrintString

    MOV DL, numTickets          ; DL = 3
    ADD DL, 30H                 ; Convert to ASCII '3'
    MOV AH, 2                   ; AH = 2 (DOS character output)
    INT 21H                     ; Display the character

    CALL PrintNewline
    LEA DX, ticketPriceMsg      ; "Price per ticket: BDT "
    CALL PrintString

    MOV AX, eventPrice          ; AX = 500
    CALL DisplayNumber          ; Display the number
```

Execution jumps to DisplayNumber procedure

assembly

```
DisplayNumber PROC
    MOV tempAX, AX              ; Save registers
    MOV tempBX, BX
    MOV tempCX, CX
    MOV tempDX, DX

    ; Clear digit array, then extract digits
    ; ...extensive digit extraction code...
    ; For value 500:
    ; digitArray = [0, 0, 5, 0, 0]

    ; Display digits, skipping leading zeros
    ; Output: "500"

    MOV DX, tempDX              ; Restore registers
    MOV CX, tempCX
    MOV BX, tempBX
    MOV AX, tempAX
    RET                         ; Return to caller
DisplayNumber ENDP
```

> Execution returns to PrintEventType after displaying "500"

assembly

```
    CALL PrintNewline
    LEA DX, extrasListMsg       ; "Extras selected: "
    CALL PrintString

    CALL DisplayExtras          ; Call procedure to list extras
```

> Execution jumps to DisplayExtras procedure

assembly

```
DisplayExtras PROC
    MOV tempAX, AX              ; Save registers
    MOV tempBX, BX
    MOV tempDX, DX

    MOV BH, 0                   ; Flag for whether any extras displayed

    CMP hasParking, 1           ; Check parking
    JNE CheckDisplayMerchandise ; Not taken (hasParking = 1)
    LEA DX, parkingSelected     ; "Parking Pass, "
    CALL PrintString
    MOV BH, 1                   ; Set flag

CheckDisplayMerchandise:
    CMP hasMerch, 1             ; Check merchandise
    JNE CheckDisplayVIP         ; Jump if not (which it isn't)
    LEA DX, merchSelected
    CALL PrintString
    MOV BH, 1
```

> hasMerch = 0, so execution jumps to CheckDisplayVIP

assembly

```
CheckDisplayVIP:
    CMP hasVIP, 1               ; Check VIP
    JNE FinishExtrasDisplay     ; Not taken (hasVIP = 1)
    LEA DX, vipSelected         ; "VIP Upgrade, "
    CALL PrintString
    MOV BH, 1                   ; Set flag

FinishExtrasDisplay:
    CMP BH, 0                   ; Check if any extras displayed
    JNE DisplayExtrasDone       ; Jump if yes (which it is)
    LEA DX, noExtrasSelected
    CALL PrintString

DisplayExtrasDone:
    MOV DX, tempDX              ; Restore registers
    MOV BX, tempBX
    MOV AX, tempAX
    RET                         ; Return to caller
DisplayExtras ENDP
```

> Execution returns after displaying "Parking Pass, VIP Upgrade, "

assembly

```
DisplaySubtotal:
    CALL PrintNewline
    LEA DX, subtotalMsg          ; "Subtotal: BDT "
    CALL PrintString

    MOV AX, subtotal             ; AX = 1850
    CALL DisplayNumber           ; Display "1850"

    CMP discount, 0              ; Check if discount applies
    JE CheckCouponDiscount       ; Not taken (discount = 277)

    CALL PrintNewline
    LEA DX, discountApplied      ; "Group discount of 15% applied!"
    CALL PrintString

    CALL PrintNewline
    LEA DX, discountMsg          ; "Group Discount (15%): BDT "
    CALL PrintString

    MOV AX, discount             ; AX = 277
    CALL DisplayNumber           ; Display "277"

CheckCouponDiscount:
    CMP couponDisc, 0            ; Check if coupon discount applies
    JE DisplayTotal              ; Not taken (couponDisc = 185)

    CALL PrintNewline
    LEA DX, couponDiscountMsg    ; "Coupon Discount (10%): BDT "
    CALL PrintString

    MOV AX, couponDisc           ; AX = 185
    CALL DisplayNumber           ; Display "185"
```

Group discount and coupon discount displayed

```assembly
DisplayTotal:
    CALL PrintNewline
    LEA DX, divider             ; "-----------------------"
    CALL PrintString

    CALL PrintNewline
    LEA DX, totalBillMsg        ; "TOTAL AMOUNT: BDT "
    CALL PrintString

    MOV AX, totalBill           ; AX = 1388
    CALL DisplayNumber          ; Display "1388"

    CALL PrintNewline
    LEA DX, divider
    CALL PrintString

    CALL PrintNewline
    LEA DX, confirmBooking      ; "Would you like to confirm your booking? (Y/N): "
    CALL PrintString

    CALL GetCharToUpper         ; Get user input
                                ; *** CUSTOMER ENTERS 'Y' ***

    CMP AL, 'Y'
    JE BookingConfirmed         ; Jump if Y (which it is)
    CMP AL, 'N'                 ; Skip this check due to jump
    JE CancelBooking

    CALL PrintNewline           ; Error handling (skipped)
    LEA DX, errorInvInput
    CALL PrintString
    JMP DisplayTotal
```

> AL = 'Y', so execution jumps to BookingConfirmed

```assembly
BookingConfirmed:
    CALL PrintNewline
    LEA DX, bookingSuccessMsg    ; "Your booking has been confirmed! Enjoy the event!"
    CALL PrintString
    JMP ExitProgram              ; Jump to exit
```

> Execution jumps to ExitProgram

```assembly
ExitProgram:
    MOV AX, 4C00H                ; AX = exit function with code 0
    INT 21H                      ; Call DOS to terminate program
```

> Program terminates successfully

Final Bill Summary

The program calculated the following values:

- Base ticket cost: 3 × 500 = 1500 BDT
- Extras: Parking (100 BDT) + VIP (250 BDT) = 350 BDT
- Subtotal: 1850 BDT
- Group discount (15%): 277 BDT
- Coupon discount (10%): 185 BDT
- **Total bill**: 1388 BDT

The program displayed all these details to the customer, received confirmation, and completed the booking successfully.