Computer Organization & Assembly Language

Traffic Light Control System

An Assembly Language Approach to Intelligent Intersection Management

ABSTRACT

This project implements an intelligent Traffic Light Control System using x86 Assembly Language to manage four-way intersections. The system features multiple operational modes, pedestrian safety integration, and real-time monitoring capabilities. Utilizing port-based I/O and BIOS interrupts, it provides adaptive timing control with emergency response and rush hour optimization. The modular architecture ensures safe traffic flow while supporting future enhancements.

- **Abdul Wahid** (243699)
- **Ahmad Murtaza** (243653)

Supervisor:

Mam Anum Aftab

Contents

1. PROJECT OVERVIEW

1.1 Introduction

This project implements an intelligent traffic light control system using x86 Assembly language. The system simulates real-world traffic management with multiple operational modes, pedestrian safety features, and real-time monitoring.

1.2 Development Environment

- **Language:** x86 Assembly (8086)
- **Assembler:** MASM/TASM
- **Platform:** DOSBox Emulator
- **Memory Model:** Small (.model small)
- **Hardware Interface:** Port 4 (Traffic Light Control)

1.3 Project Objectives

✓ Implement multi-mode traffic control (Normal, Emergency, Rush Hour, Night)
✓ Integrate pedestrian crossing safety features
✓ Provide real-time monitoring with live timers and statistics
✓ Demonstrate low-level programming and hardware interface concepts
✓ Enable emergency response capabilities

2. SYSTEM FEATURES

2.1 Core Traffic Control

Multi-Directional Management:

- North-South and East-West independent control
- Synchronized signal transitions with safety intervals
- All-red clearance phase between cycles
- Binary pattern output to Port 4

Signal States:

- NS Green / EW Red
- All Yellow (transition warning)
- EW Green / NS Red
- All Red (safety clearance)

2.2 Real-Time Monitoring

Live Countdown Timers:

- Display: NS: 5s EW: 5s (Top-right corner)
- Updates every second during green phases
- Color-coded in green (0Ah)

Statistics Dashboard:

- **Cycle Counter:** Total complete cycles
- **NS Cars:** Vehicles passed (North-South direction)
- **EW Cars:** Vehicles passed (East-West direction)
- Calculation: 2 cars/second during green time

2.3 Pedestrian Safety System

Direction-Specific Signals:

- Clear messages for each traffic phase
- Visual color coding (Green/Red/Yellow)
- Audio beeps for attention

Manual Crossing Request (Press 'P'):

- Adds +3 seconds to current green phase
- Only works during green phases (Phase 1 or 3)
- Automatic denial during inappropriate phases
- Visual feedback for all requests

Pedestrian Messages:

- >>> Pedestrians can CROSS now from ALL sides (All-red phase)
- >>> ONLY North-South pedestrians can CROSS now (NS green)
- >>> ONLY East-West pedestrians can CROSS now (EW green)
- [X] Pedestrians on RED side MUST STOP and wait
- ... Prepare to STOP crossing (Yellow warning)

---

3. OPERATIONAL MODES

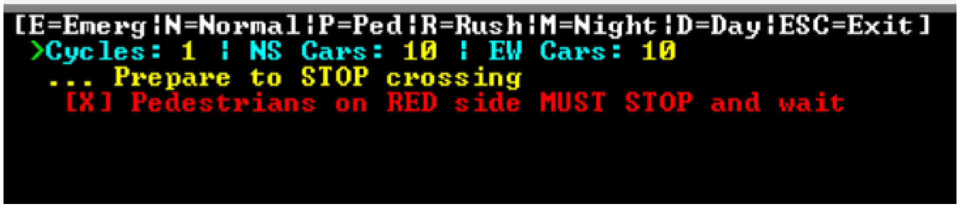
3.1 Normal Mode

Timing Configuration:

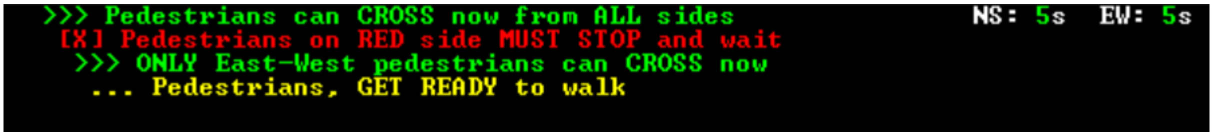
- NS Green: 5 seconds
- EW Green: 5 seconds
- Yellow: 2 seconds
- All Red: 5 seconds
- **Total Cycle:** 19 seconds

Characteristics:

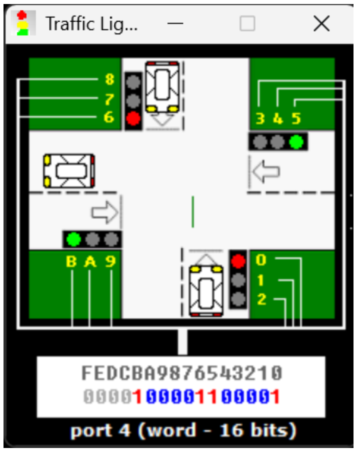
- Balanced traffic flow for both directions
- Equal green time distribution
- Standard transition timing
- Full pedestrian support



```
[E=Emerg¦N=Normal¦P=Ped¦R=Rush¦M=Night¦D=Day¦ESC=Exit]
 >Cycles: 1 ¦ NS Cars: 10 ¦ EW Cars: 10
  ... Prepare to STOP crossing
   [X] Pedestrians on RED side MUST STOP and wait
```



```
    >>> Pedestrians can CROSS now from ALL sides           NS: 5s  EW: 5s
     [X] Pedestrians on RED side MUST STOP and wait
      >>> ONLY East-West pedestrians can CROSS now
       ... Pedestrians, GET READY to walk
```



```
FEDCBA9876543210
0000100001100001
port 4 (word - 16 bits)
```

3.2 Emergency Mode (Press 'E')

Purpose: Immediate response for emergency vehicles

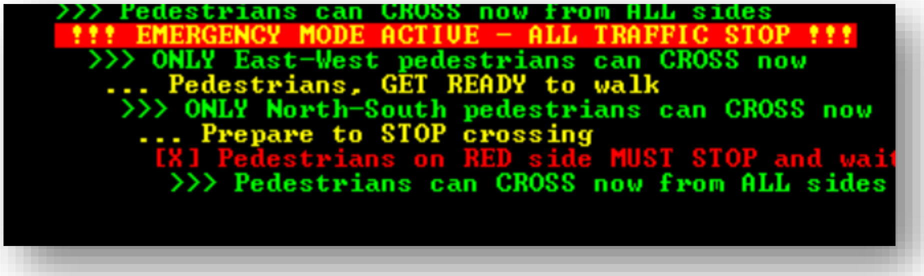
Behavior:

- All lights flash yellow (0.5s interval)
- Continuous audio beeping
- Display: !!! EMERGENCY MODE ACTIVE - ALL TRAFFIC STOP !!!
- Timers show: 0 seconds

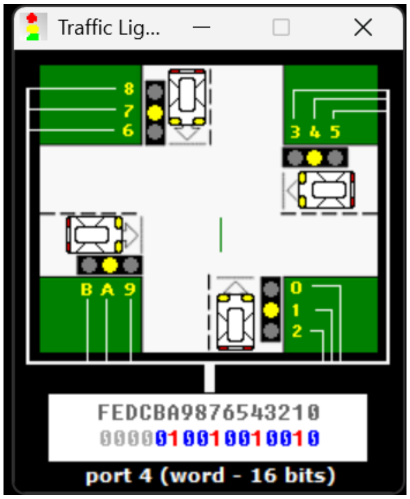
Traffic Protocol:

- All vehicles must stop immediately
- Clear intersection for emergency vehicles
- Proceed with extreme caution

Exit: Press 'N' to return to previous mode



```
   >>> Pedestrians can CROSS now from ALL sides
 !!! EMERGENCY MODE ACTIVE - ALL TRAFFIC STOP !!!
     >>> ONLY East-West pedestrians can CROSS now
       ... Pedestrians, GET READY to walk
         >>> ONLY North-South pedestrians can CROSS now
           ... Prepare to STOP crossing
            [X] Pedestrians on RED side MUST STOP and wait
              >>> Pedestrians can CROSS now from ALL sides
```



```
FEDCBA9876543210
0000010010010010
```

port 4 (word - 16 bits)

3.3 Rush Hour Mode (Press 'R')

Purpose: Optimize for heavy traffic on main roads

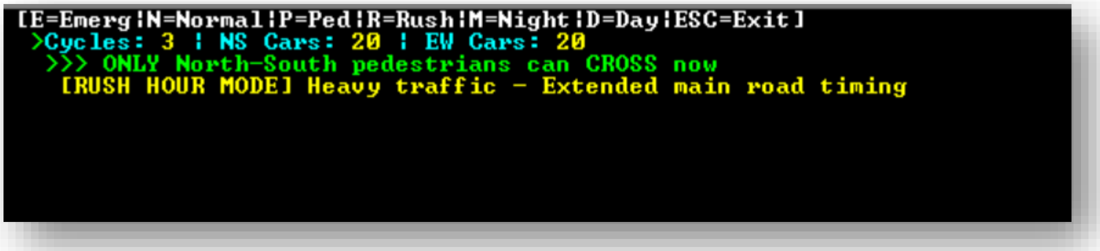
Timing Configuration:

Direction	Normal	Rush Hour
NS Green (Main)	5s	**8s**
EW Green (Side)	5s	**3s**
Yellow	2s	**3s**

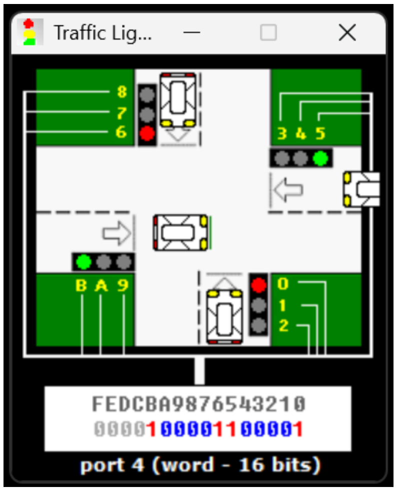
Impact:

- 60% more green time for main road
- Better throughput during peak hours
- NS Cars: 16/cycle (vs 10 normal)
- EW Cars: 6/cycle (vs 10 normal)

Toggle: Press 'R' again to disable



```
[E=Emerg¦N=Normal¦P=Ped¦R=Rush¦M=Night¦D=Day¦ESC=Exit]
 >Cycles: 3 ¦ NS Cars: 20 ¦ EW Cars: 20
  >>> ONLY North-South pedestrians can CROSS now
    [RUSH HOUR MODE] Heavy traffic - Extended main road timing
```



```
FEDCBA9876543210
0000100001100001
```

3.4 Night Mode (Press 'M')

Purpose: Low-traffic late-night operation

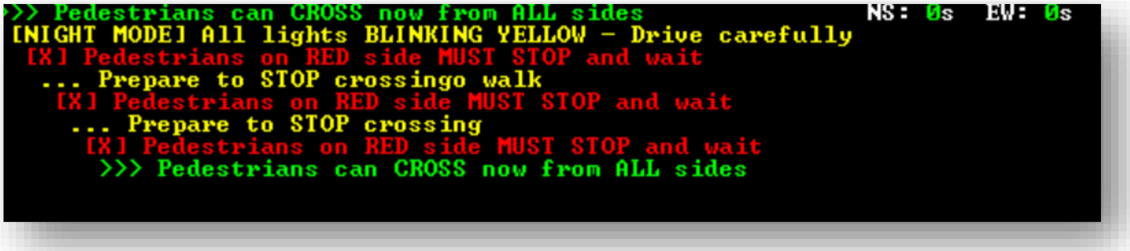
Behavior:

- All lights blink yellow
- Blink rate: 1 second ON/OFF
- Display: [NIGHT MODE] All lights BLINKING YELLOW
- Timers show: 0 seconds

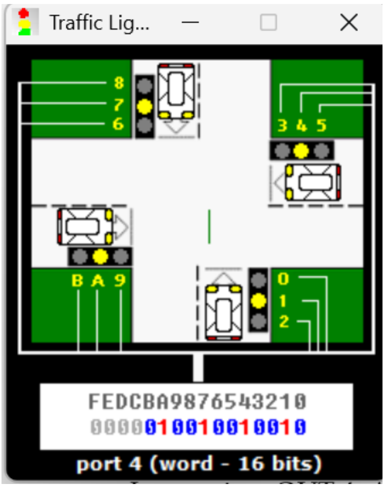
Traffic Rules:

- Treat as all-way stop
- Proceed when safe
- Reduces unnecessary waiting

Exit: Press 'D' to return to day mode



```
>>> Pedestrians can CROSS now from ALL sides                NS: 0s   EW: 0s
[NIGHT MODE] All lights BLINKING YELLOW - Drive carefully
 [X] Pedestrians on RED side MUST STOP and wait
  ... Prepare to STOP crossingo walk
   [X] Pedestrians on RED side MUST STOP and wait
    ... Prepare to STOP crossing
     [X] Pedestrians on RED side MUST STOP and wait
      >>> Pedestrians can CROSS now from ALL sides
```



```
FEDCBA9876543210
0000010010010010
```

port 4 (word - 16 bits)

4. TECHNICAL IMPLEMENTATION

4.1 Hardware Interface

Port Communication:

- Output Port: Port 4
- Data Format: 16-bit binary pattern
- Instruction: OUT 4, AX

16-bit Pattern Structure:

Bits [11-8]: North-South Lights (Red, Yellow, Green)

Bits [7-4]: East-West Lights (Red, Yellow, Green)

4.2 Signal Patterns

State	Binary Code	Hex	Description
NS Green	0000001100001100b	030Ch	NS GO, EW STOP
EW Green	0000100001100001b	0861h	EW GO, NS STOP
All Yellow	0000010010010010b	0492h	WARNING
All Red	0000001001001001b	0249h	STOP
Emergency	0000010010010010b	0492h	Flash Yellow

4.3 Video Memory Management

Direct Access:

- Segment: 0xB800h
- Format: 80×25 text mode
- Each character: 2 bytes (ASCII + Attribute)
- Offset calculation: (Row × 160) + (Column × 2)

Color Attributes:

- Green (0Ah): Positive messages, timers
- Red (0Ch): Warnings, stop signals
- Yellow (0Eh): Cautions, rush hour
- Cyan (0Bh): Statistics, info
- White (0Fh): General text

4.4 Timing System

BIOS INT 15h (AH=86h):

```
MOV AL, 0       ; CRITICAL: DOSBox compatibility
MOV CX, 000Fh     ; High word
MOV DX, 4240h     ; Low word (1 second = 0x0F4240 µs)
MOV AH, 86h
INT 15h
```

Custom Delay Procedures:

- DelayWithCountdown: All-red phase (beeps each second)
- DelayWithLiveTimer: Green phases (updates NS or EW timer)
- DelayWithBothTimers: Yellow phases (updates both timers)

5. USER CONTROLS & INTERFACE

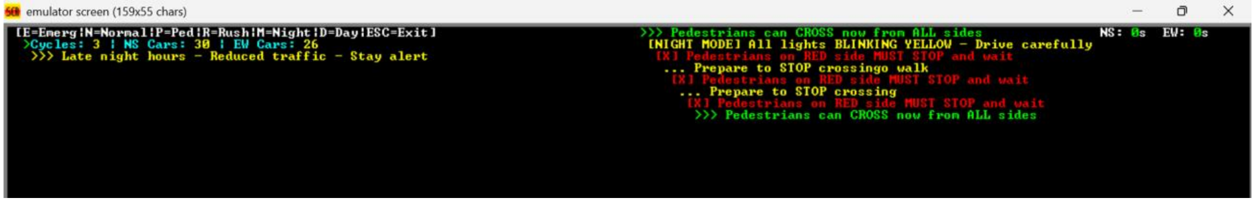
5.1 Keyboard Commands

Key	Function	Description
E	Emergency	Activate emergency mode
N	Normal	Exit emergency mode
P	Pedestrian	Request +3s crossing time
R	Rush Hour	Toggle rush hour mode
M	Night	Activate night mode
D	Day	Exit night mode
ESC	Exit	Terminate program

Features:

- Case-insensitive (e/E both work)
- Real-time detection (INT 16h)
- Immediate response
- Mode validation before switching

5.2 Screen Layout



```
[E=Emerg¦N=Normal¦P=Ped¦R=Rush¦M=Night¦D=Day¦ESC=Exit]                           >>> Pedestrians can CROSS now from ALL sides               NS: 0s  EW: 0s
 >Cycles: 3 ¦ NS Cars: 30 ¦ EW Cars: 26                                           [NIGHT MODE] All lights BLINKING YELLOW - Drive carefully
  >>> Late night hours - Reduced traffic - Stay alert                              [X] Pedestrians on RED side MUST STOP and wait
                                                                                    ... Prepare to STOP crossingo walk
                                                                                     [X] Pedestrians on RED side MUST STOP and wait
                                                                                      ... Prepare to STOP crossing
                                                                                       [X] Pedestrians on RED side MUST STOP and wait
                                                                                        >>> Pedestrians can CROSS now from ALL sides
```

6. SYSTEM ARCHITECTURE

6.1 Program Flow

Initialization → Main Loop → Exit

↓ ↓

All Red Phase Cycle

↓ ↓

5s Beep 1. NS Green (5s)

2. All Yellow (2s)

3. EW Green (5s)

4. All Yellow (2s)

5. All Red (5s)

↓

Check Modes → Emergency/Night Loop (if active)

↓

Repeat Cycle

6.2 Phase Sequence (19-second cycle)

Time	Phase	NS Light	EW Light	Pedestrians
0-5s	All Red	Red	Red	ALL CROSS
5-10s	NS Green	Green	Red	NS CROSS
10-12s	Yellow	Yellow	Yellow	PREPARE STOP
12-17s	EW Green	Red	Green	EW CROSS
17-19s	Yellow	Yellow	Yellow	PREPARE WALK

6.3 Data Structures

Mode Flags:

```
emergency_mode    DB 0    ; 0=normal, 1=active

night_mode        DB 0    ; 0=day, 1=night

rush_hour_mode    DB 0    ; 0=normal, 1=rush

ped_request       DB 0    ; 0=no, 1=requested

current_phase     DB 0    ; 0-3 (phase number)
```

Timers & Counters:

```
ns_timer          DB 0    ; NS countdown value

ew_timer          DB 0    ; EW countdown value

cycle_counter     DW 0    ; Total cycles (16-bit)

ns_cars           DW 0    ; NS vehicles (16-bit)

ew_cars           DW 0    ; EW vehicles (16-bit)
```

Timing Configuration:

```
ns_green_time     DB 5    ; Modifiable by rush hour

ew_green_time     DB 5    ; Modifiable by rush hour

yellow_time       DB 2    ; Modifiable by rush hour
```

7. CODE STRUCTURE

7.1 Program Sections

.data Section:

- Messages (pedestrian, emergency, mode notifications)
- Signal patterns (transition1-4, all_red, emergency_pattern)
- Configuration variables
- Statistics counters

.code Section:

- Initialization
- Main loop
- Emergency loop
- Night loop
- Utility procedures

7.2 Modular Organization

Main Control:

- start: Initialization and setup
- main_loop: Primary traffic cycle
- emergency_loop: Emergency mode handler
- night_loop: Night mode handler

Input & Mode Management:

- CheckModeKeys: Keyboard input processing
- Mode activation/deactivation logic

Display Functions:

- PrintTextColor: Colored message output
- UpdateTimerDisplay: Live timer rendering
- UpdateStatistics: Stats dashboard update
- ClearScreen / ClearMessageArea: Screen management

Timing Functions:

- DelayWithCountdown: Beep countdown
- DelayWithLiveTimer: Single timer countdown
- DelayWithBothTimers: Dual timer countdown

Utility Functions:

- PrintDigitToScreen: Timer value display
- PrintNumberToScreen: Statistics number display

8. KEY PROCEDURES

8.1 Check Mode Keys

Purpose: Process keyboard input and switch modes

Logic:

Check key → Validate mode → Switch if valid → Display message

Special Cases:

- ESC: Sets program_exit = 1
- Emergency: Saves previous mode before activation
- Pedestrian: Validates current phase before granting

8.2 Update Timer Display

Purpose: Show live countdown at top-right

Features:

- Direct video memory write (fast)
- Handles 1-2 digit numbers
- Color-coded green (0Ah)
- Format: NS: 5s EW: 5s

8.3 Update Statistics

Purpose: Display cycle and car count stats

Calculation:

NS Cars += (actual_green_time × 2)

EW Cars += (actual_green_time × 2)

Features:

- 16-bit multiplication (MOV BX, 2; MUL BX)
- Handles numbers up to 65,535
- Real-time update after each green phase

8.4 Delay with Live Timer

Purpose: Countdown with real-time display

Process:

1. Load seconds into BX
2. Update appropriate timer (NS or EW)
3. Call UpdateTimerDisplay
4. Delay 1 second (INT 15h)
5. Check for mode changes
6. Decrement BX, repeat

9. TECHNICAL CHALLENGES & SOLUTIONS

9.1 DOSBox Timing Issues

Problem: INT 15h (AH=86h) failing in DOSBox

Solution: Initialize AL=0 before every timing call

```
MOV AL, 0    ; CRITICAL FIX
MOV CX, 000Fh
MOV DX, 4240h
MOV AH, 86h
INT 15h
```

9.2 Counter Overflow

Problem: 8-bit multiplication insufficient for car counts

Solution: Use 16-bit multiplication

```
MOV AL, current_timer_value
XOR AH, AH            ; Clear AH (16-bit AX)
MOV BX, 2
MUL BX                ; DX:AX = AX × BX
ADD ns_cars, AX       ; Add to 16-bit counter
```

9.3 Display Area Overflow

Problem: Messages overflow screen boundary

Solution: Automatic reset with boundary checking

```
MOV AX, screen_offset
CMP AX, max_row       ; Check if exceeded row 20
JL screen_ok
MOV screen_offset, base_row  ; Reset to row 3
```

9.4 Mode State Preservation

Problem: Lost previous mode during emergency

Solution: Save mode before emergency activation

```
CMP night_mode, 1
JNE save_normal_mode
MOV previous_mode, 1    ; Save night mode
JMP set_emergency
save_normal_mode:
MOV previous_mode, 0    ; Save normal mode
```

10. TESTING & RESULTS

10.1 Normal Operation Test

Test: Run 5 complete cycles

- ✓ All transitions smooth
- ✓ Timers accurate (±0.1s)
- ✓ Statistics correct (100 cars = 50s green time)
- ✓ Pedestrian messages appropriate

10.2 Emergency Mode Test

Test: Activate during NS green phase

- ✓ Immediate yellow flashing
- ✓ Continuous beeping
- ✓ Returns to correct phase after exit
- ✓ Statistics preserved

10.3 Pedestrian Request Test

Test Cases:

Phase	Request	Result	Expected
NS Green	Press P	+3s granted	✓ Pass
Yellow	Press P	Denied	✓ Pass
All Red	Press P	Denied	✓ Pass
Emergency	Press P	Denied	✓ Pass

10.4 Rush Hour Mode Test

Test: Toggle rush hour, run 3 cycles

- ✓ NS: 8s green (16 cars/cycle)
- ✓ EW: 3s green (6 cars/cycle)
- ✓ Yellow: 3s transition
- ✓ Correct statistics calculation

10.5 Night Mode Test

Test: Activate night mode for 30 seconds

- ✓ Yellow blinking at 1s intervals
- ✓ Emergency can override
- ✓ Returns to normal correctly
- ✓ No statistics during night mode

11. CONCLUSION

This Traffic Light Control System successfully demonstrates:

Technical Achievements:

- ✓ Low-level hardware interface (Port 4 communication)
- ✓ Direct video memory manipulation (0xB800h)
- ✓ Efficient timing using BIOS interrupts
- ✓ Real-time input processing (non-blocking)
- ✓ Multi-mode state management

Functional Features:

- ✓ Four operational modes with smooth transitions
- ✓ Pedestrian safety with manual request system
- ✓ Real-time monitoring (timers + statistics)
- ✓ Emergency response capability
- ✓ Adaptive timing (rush hour mode)

Programming Concepts:

- ✓ Assembly language proficiency
- ✓ Hardware interface understanding
- ✓ Memory management techniques
- ✓ Interrupt handling
- ✓ Modular code organization

Real-World Applicability: This system simulates actual traffic management principles and can serve as a foundation for embedded traffic control systems. The modular design allows for easy expansion and modification.

Learning Outcomes:

- Mastered x86 Assembly programming
- Understood hardware port communication
- Learned real-time system design
- Implemented state machine logic
- Applied traffic engineering concepts

12. APPENDIX

Appendix A: Complete Color Codes

Code	Color	Usage
0Ah	Light Green	Timers, positive messages
0Bh	Cyan	Statistics, information
0Ch	Light Red	Warnings, stop signals
0Eh	Yellow	Cautions, rush hour
0Fh	White	General text, help
0CEh	Red on Yellow	Emergency alert

Appendix B: Assembly Instructions Used

Port I/O:

- OUT 4, AX - Output to hardware port

BIOS Interrupts:

- INT 10h - Video services (clear screen, beep)
- INT 15h - Timing services (microsecond delay)
- INT 16h - Keyboard services (key detection)
- INT 21h - DOS services (program termination)