PROJECT 4

BCD CLOCK & HLL INTERFACE

CSCI 150
ASSEMBLY LANGUAGE

MAI PHAM

DEVELOPMENT ENVIRONMENT
MSVS - 2017

TABLE OF CONTENTS

- Project Status
- Input/Output Results
- Source Code (addem.asm/main.cpp)

PROJECT STATUS:

Objective

- Create a high level language (C++) that call an assembly language file to set clock and advance clock by one second.

Status/Extra Credit

- The project is completed and successfully run the main project as well as the extra credit to get current time. Note, I did make a few modification for the project so it easier for me to implement. For example, I change the structureInfor into hour, min, and sec instead of sec, min, and hour. I also put in a menu and some while loop so it easier for me to do multiple test cases at the same time.

INPUT/OUTPUT RESULTS:

Menu
1. Enter the time
2. Get current time
3. Quit
Please enter an option --> 1

Enter the time: 11:59:56A
Current Time: 11:59:56A
Enter seconds to advance: 3600
Current Time: 12:59:56P

Menu
1. Enter the time
2. Get current time
3. Quit
Please enter an option --> 1

Enter the time: 11:59:56P
Current Time: 11:59:56P
Enter seconds to advance: 86400
Current Time: 05:59:56P

Menu
1. Enter the time
2. Get current time
3. Quit
Please enter an option --> 1

Enter the time: 11:59:59P
Current Time: 11:59:59P
Enter seconds to advance: 31
Current Time: 12:00:30A

Menu
1. Enter the time
2. Get current time
3. Quit
Please enter an option --> 1

Enter the time: 12:59:45A
Current Time: 12:59:45A
Enter seconds to advance: 30
Current Time: 01:00:15A

Menu
1. Enter the time
2. Get current time
3. Quit
Please enter an option --> 1

Enter the time: 02:13:59A
Current Time: 02:13:59A
Enter seconds to advance: 22
Current Time: 02:14:21A

Menu
1. Enter the time
2. Get current time
3. Quit
Please enter an option --> 2

Current Time: 11:26:36A
Enter seconds to advance: 60
Current Time: 11:27:36A

Menu
1. Enter the time
2. Get current time
3. Quit
Please enter an option --> 2

Current Time: 11:26:48A
Enter seconds to advance: 3600
Current Time: 12:26:48P

Menu
1. Enter the time
2. Get current time
3. Quit
Please enter an option --> 2

Current Time: 11:26:53A
Enter seconds to advance: 7200
Current Time: 01:26:53P

Menu
1. Enter the time
2. Get current time
3. Quit

Please enter an option --> 3
Press any key to continue . . .

SOURCE CODE:
Main.cpp

```
#include <iostream>
#include <iomanip>
using namespace std;

struct TimeInfor {
	unsigned int hour, min, sec;
	char amPm;
};

void interpretTime(struct TimeInfor *tmPtr, const char *time) {
	int i;
	// strtol
	tmPtr->hour = time[1] - '0';
	i = (time[0] - '0') * 10;
	tmPtr->hour += i;

	tmPtr->min = time[4] - '0';
	i = (time[3] - '0') * 10;
	tmPtr->min += i;

	tmPtr->sec = time[7] - '0';
	i = (time[6] - '0') * 10;
	tmPtr->sec += i;

	tmPtr->amPm = time[8];
}

void printClock(const char clock[])	{
	cout << "Current Time: " << setw(2) << setfill('0') << hex << int(clock[0]);
	cout << ":" << setw(2) << setfill('0') << hex << int(clock[1]);
	cout << ":" << setw(2) << setfill('0') << hex << int(clock[2]);
	cout << clock[3] << endl;
}

//extern "C" int addem(int p1, int p2, int p3);
extern "C" {
	void setClock(char clock[], const struct TimeInfor *tmPtr);
	void tickClock(char clock[]);
	void getCurrentTime(char time[]);
	//unsigned char incrementClockValue(char BCDbits, const unsigned int maxValue);

	long IndexOf(long n, long array[], unsigned count);
	// Assembly language module
}

int main()	{
	//TimeInfor *tmPtr = new TimeInfor;

	TimeInfor tmPtr;
	int s;
	char time[30];
```

```
	char clock[5];
	/*getCurrentTime(time);
	cout << hex << int(time[0]) << hex << int(time[1]) << endl;
	cout << hex << int(time[2]) << hex << int(time[3]) << endl;
	cout << hex << int(time[4]) << hex << int(time[5]) << endl;
	interpretTime(&tmPtr, time);
	cout << int(tmPtr.hour) << endl;
	cout << int(tmPtr.min) << endl;
	cout << int(tmPtr.sec) << endl;
	cout << tmPtr.amPm << endl;
	setClock(clock, &tmPtr);
	printClock(clock);
	tickClock(clock);
	printClock(clock);
	*/

	int option;
	cout << "Menu\n";
	cout << "1. Enter the time\n";
	cout << "2. Get current time\n";
	cout << "3. Quit\n";
	cout << "Please enter an option --> ";
	cin >> option;
	while (option != 3) {
		if (option == 1) {
			cout << "\nEnter the time: ";
			cin >> time;
		}
		else if (option == 2) {
			getCurrentTime(time);
			cout << endl;
		}
		interpretTime(&tmPtr, time);
		setClock(clock, &tmPtr);
		printClock(clock);
		cout << "Enter seconds to advance: ";
		cin >> s;
		for (int i = 0; i < s; i++)
			tickClock(clock);
		printClock(clock);
		cout << "\nMenu\n";
		cout << "1. Enter the time\n";
		cout << "2. Get current time\n";
		cout << "3. Quit\n";
		cout << "Please enter an option --> ";
		cin >> option;
	}
	return 0;
}
```

addem.asm

```
; The addem Subroutine    (addem.asm)
; This subroutine links to Visual C++.

;.386P
;.model flat
;public _addem
```

```
INCLUDE Irvine32.inc

setClock PROTO C,
	clock:PTR BYTE, arrayPtr:PTR DWORD
tickClock PROTO C,
	clock:PTR BYTE
getCurrentTime PROTO C,
	time:PTR BYTE

.code
setClock PROC C, clock:PTR BYTE, structPtr:PTR DWORD
	pushad
	mov esi, structPtr
	mov edi, clock
	mov ecx, 3

L1:
	mov eax, 0
	mov edx, 0
	mov eax, [esi]			; eax = struct value
	mov ebx, 10
	div ebx
	shl al, 4			; mov 1st digit to upper al
	or al, dl			; mov 2rd digit to lower al
	mov [edi], al			; store in clock
	add esi, 4
	inc edi
	loop L1

	mov al, [esi]			; store AM/PM
	mov [edi], al
	popad
	ret
setClock ENDP

tickClock PROC C clock:PTR BYTE
	pushad
	mov eax, 0
	mov ebx, 0
	mov esi, clock
	add esi, 2			; go to seconds index
	mov ecx, 3

L1:
	mov al, [esi]
	mov bl, [esi]
	and al, 11110000b		; keep upper half in case
	and bl, 00001111b		; keep lower half
	inc bl
	cmp ecx, 1			; check if in hour index,
	je hour					;   if yes, go to hour loop
	cmp bl, 10			; if min/sec < 10,
	jb done					;   store the value
	shr al, 4			; if above, shift and inc
	inc al				;   the upper half
	cmp al, 6			; if upper half < 6
	jb almostDone			;   store upper half
```

```
	mov dl, 0			; else store 0 and do it again
	mov [esi], dl
	dec esi
	loop L1

hour:
	cmp bl, 2			; if hour < 2, store it
	jb done
	cmp bl, 3
	ja done				; if hour > 3, store it
	shr al, 4			; else, check to see if 02 or 12
	cmp al, 0
	je done				; if 02/03, store the value
	cmp bl, 3
	je finish			; if 13, change to 1 by storing the 1 only
	shl al, 4			; else keep upper half and
	mov edi, clock			; check for am/pm
	mov dl, [edi+3]
	cmp dl, 50h
	je am				; if pm, change to am
	cmp dl, 41h
	je pm				; if am, change to pm
am:
	mov dl, 41h			; change to am
	mov [edi+3], dl
	jmp done
pm:
	mov dl, 50h			; change to pm
	mov [edi+3], dl
	jmp done
almostDone:
	shl al, 4			; mov upper half left
	jmp finish
done:
	or al, bl			; combine 2 digits
finish:
	mov [esi], al			; store the value
	popad
	ret
tickClock ENDP

getCurrentTime PROC C, time:PTR BYTE
	pushad
	mov eax, 0
	mov edx, 0
	mov esi, time
	mov al, 41h
	mov [esi+8], al			; store AM 1st


	call getMseconds		; get current time
	mov ebx, 1000			; get ricks of milli seconds
	div ebx
	mov edx, 0
	mov ebx, 3600			; eax = hour, edx = min+sec
	div ebx
	cmp al, 10			; hr < 10, store it
	jb l1
```

```
        cmp al, 12                      ; hr > 12, min 12 and change to pm
        ja l2
        jmp l3                          ; else divide and store
L2:
        sub ax, 12                      ; change to 12 hr format
        mov bl, 50h                     ; cahnge to pm
        mov [esi+8], bl
L1:
        mov bl, 0                       ; store leading 0
        mov [esi], bl
        mov [esi + 1], al               ; store lower half
        jmp next
L3:
        push edx                        ; save min/sec
        mov bl, 10                      ; slit two digits
        div bl
        mov [esi], al                   ; store upper half
        mov [esi + 1], ah               ; store lower half
        pop edx                         ; restore min/sec
next:
        mov eax, edx                    ; get min/sex
        mov ebx, 60
        mov edx, 0
        div ebx                         ; eax = min, edx = sec

        mov bl, 10                      ; split two digits
        div bl
        mov [esi + 3], al               ; store upper digit
        mov [esi + 4], ah               ; store lower digit

        mov eax, edx                    ; get sec
        mov bl, 10                      ; split two digit
        mov ah, 0
        div bl
        mov [esi + 6], al               ; store upper digit
        mov [esi + 7], ah               ; store lower digit

        mov eax, 0
        mov esi, time
        mov ecx, 8                      ; convert all digit to char
L10:
        mov al, [esi]
        add al, 30h
        mov [esi], al
        inc esi
        loop l10

        popad
        ret
getCurrentTime ENDP

END
```