CMPE-013/L

Introduction to "C" Programming

Max Lichtenstein

Piazza Poll: Which one(s) print "!olleH" when passed "Hello!"?

```c
char rPrint_A(char * str)
{
    if (*str == NULL) {
        return *(str);
    } else {
        char nextChar = rPrint_A(str + 1);
        printf("%c", nextChar);
        return *(str);
    }
}

void rPrint_B(char * str)
{
    if (*str == NULL) {
        return;
    } else {
        rPrint_B(str + 1);
        printf("%c", *(str));
        return;
    }
}
```

```c
char rPrint_C(char * str)
{
    if (*str == NULL) {
        return *(str-1);
    } else {
        printf("%c", *(str));
        char nextChar = rPrint_C(str + 1);
        return nextChar;
    }
}

char rPrint_D(char * str)
{
    if (*str == NULL) {
        return *(str-1);
    } else {
        char nextChar = rPrint_D(str + 1);
        printf("%c", nextChar);
        return *(str-1);
    }
}
```

Roadmap

- Announcements, Lab 08 notes
- Piazza Poll ✓
- Binary Trees + Recursion
  - Work group time
  - Class coding time
- BREAK
- Lab08:
  - Overview
  - Decoding
  - Events with Parameters, Local events
  - Timers
- Maximum timer speed (?)
- Lab 9 stuff (?)

Announcements:

- BattleBoats Partner Selection
  - DO IT
- New Battleboats Late Hours policy

  MAX (late hours A, late hours B)
- Lab10 ✓
  - design_goals.txt due next Tuesday //
    - Start a convo on Piazza

Announcements: Grade breakdown

Lab08 Fixes

- So many changes:
  - Lab08_main.c, not lab08_main.c
    - OLEDPrint(MorseEvent event), not OLEDPrint(morseEvent event)
  - Added missing files to .zip
    - All the oled/oleddriver/buttons stuff
    - MorseTest_template.h
  - Morse.h:
    - Changed documentation for MorseDecode (should return ERROR events when attempting to decode invalid morse strings)
    - Should only have ONE MorseEventType enum
  - Lab manual :
    - Test files *are* required
    - Fixed typo in MorseCheckEvents state machine

```c
char rPrint_A(char * str)
{
    if (*str == NULL) {
        return *(str);
    } else {
        char nextChar = rPrint_A(str + 1);
        printf("%c", nextChar);
        return *(str);
    }
}
```

"!olle"

???

'e'

H

rprintA("Hello!")

↳ rprintA("ello\0")

↳ rprint("!\0")

↳ rprint("\0")

'\0'

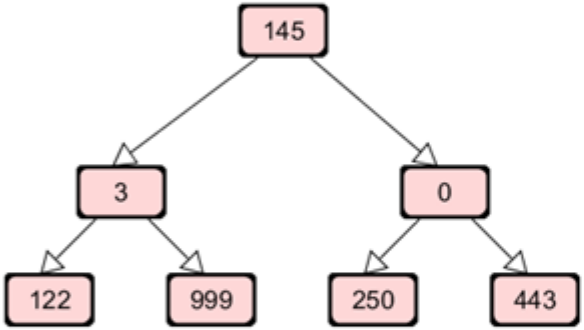
```c
void rPrint_B(char * str)
{
    if (*str == NULL) {
        return;
    } else {
        rPrint_B(str + 1);
        printf("%c", *(str));
        return;
    }
}
```

```c
char rPrint_C(char * str)
{
    if (*str == NULL) {
        return *(str-1);
    } else {
        printf("%c", *(str));
        char nextChar = rPrint_C(str + 1);
        return nextChar;
    }
}
```

```c
char rPrint_D(char * str)
{
    if (*str == NULL) {
        return *(str-1);
    } else {
        char nextChar = rPrint_D(str + 1);
        printf("%c", nextChar);
        return *(str-1);
    }
}
```

Recursive Binary Trees

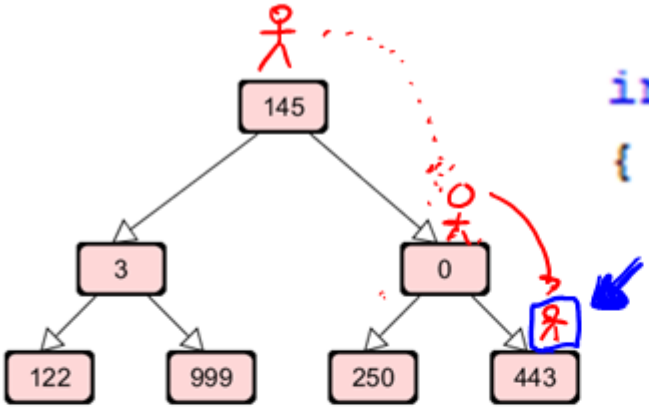
Simple Binary Tree Problem: Find Maximum Value in Tree




```
typedef struct Node {
    struct Node *leftChild;
    struct Node *rightChild;
    int data;
} Node;

int findMax(Node* ~~root~~ node);
```

```
printf("Largest number in tree = %d\n", findMax(root));
```

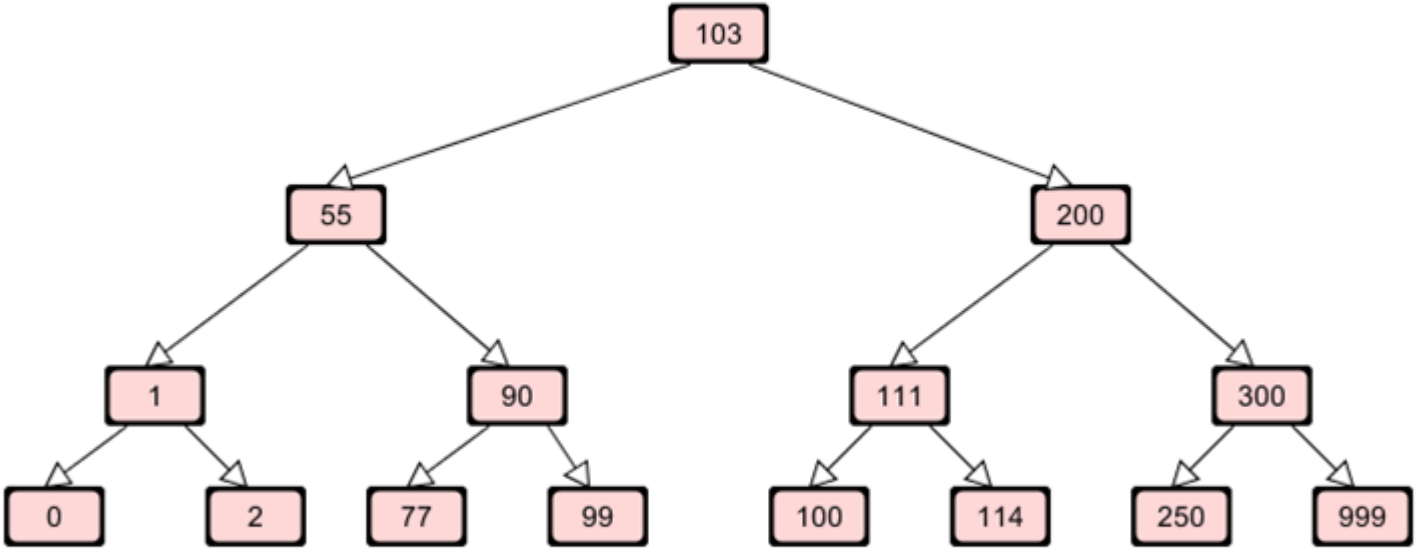


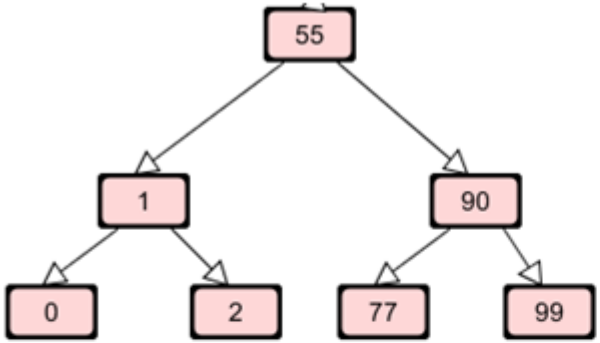

```
int findMax(Node* root)
{
```

```
findMax ( Node* node){
//base case:
    if (RC is Null || LC is Null):
        return node->data;
    return Max3(
        findMax( Right(node));
        find Max (Left(node)); *
        node->data)
```

Binary tree rule:
No node can have a value greater than any node to its right anywhere in the tree

Objective: given X, find the LARGEST value in the list that is LESS THAN X






```c
typedef struct Node {
    struct Node *leftChild;
    struct Node *rightChild;
    int data;
} Node;

int best_lt(Node* root, int X)
{
    if (????){ //base case
        ?????;
        return ????;
    } else {
        ?????;
        return ????;
    }
}
```

Objective:
given X, find largest value in tree that is less than X

```c
printf("Largest number less than 10 = %d\n", best_lt(root, 10));
printf("Largest number less than 88 = %d\n", best_lt(root, 88));
printf("Largest number less than 55 = %d\n", best_lt(root, 55));
printf("Largest number less than 98 = %d\n", best_lt(root, 98));
```

BREAK

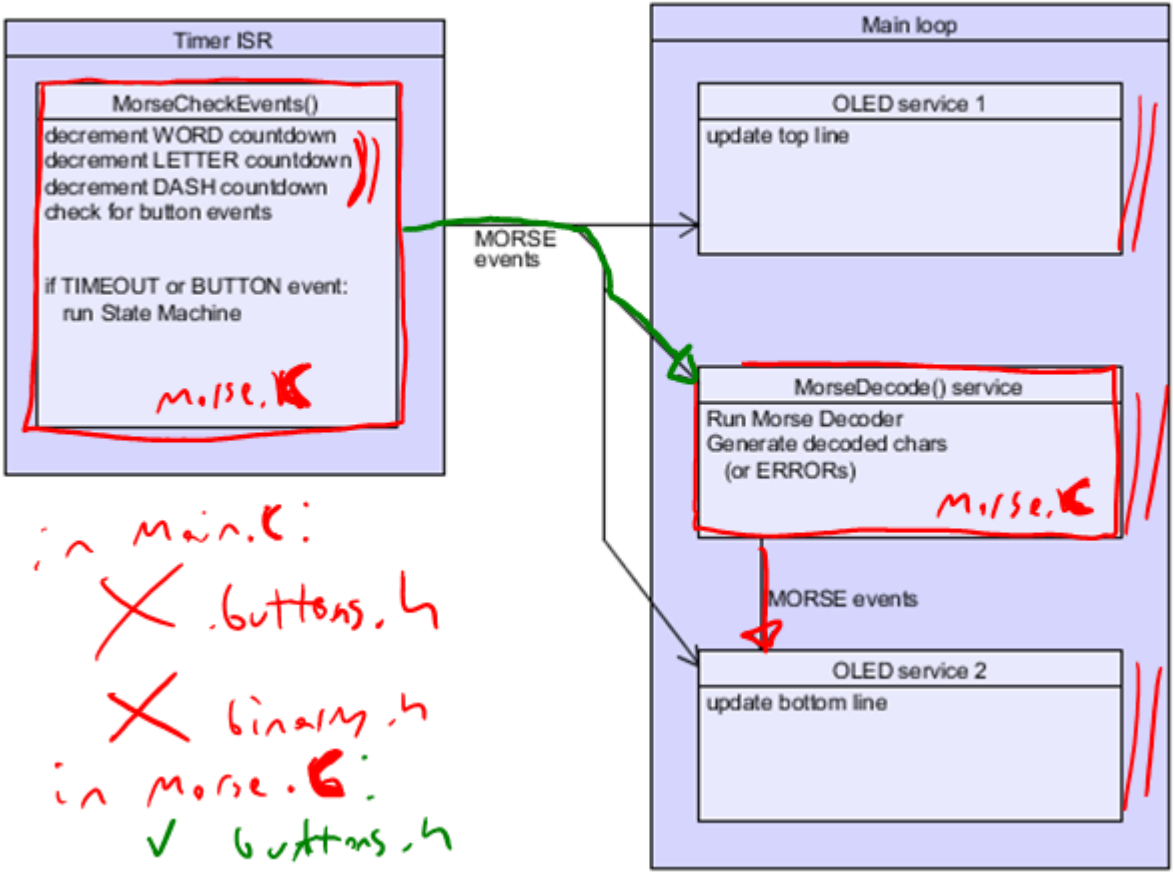
Lab 08

Lab08 overview

- Morse.h:
  - MorseInit/MorseDecode
  - MorseCheckEvents
- Lab08_main.c
  - Run MorseCheckEvents
  - Feed results to MorseDecode and print services
    - Feed *those* results to print services
- Binary tree.h
  - make tree
  - print tree

Don't get stuck
Morse Init can use
a hardcoded tree

Lab08 Big Picture



in main.c:

✗ buttons.h

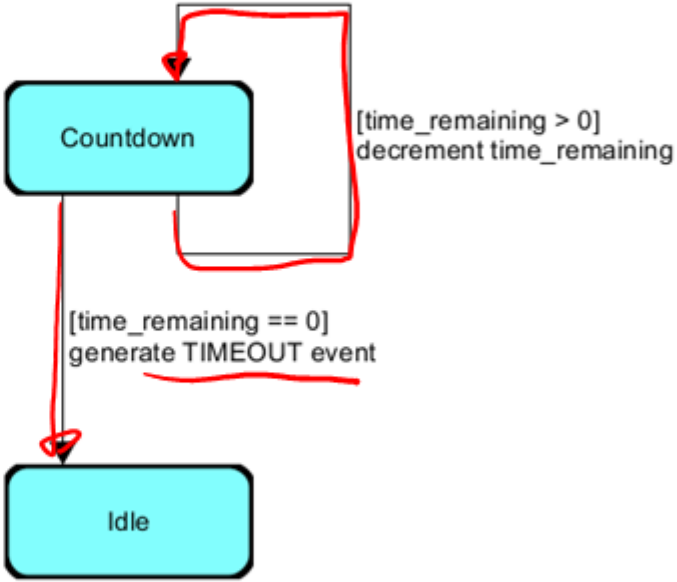
✗ binary.h

in morse.c:

✓ buttons.h

✓ binary tree.h

Countdown Timers:

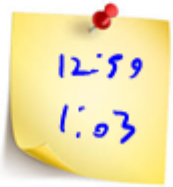


Countdown Timers vs Freerunning Timer:

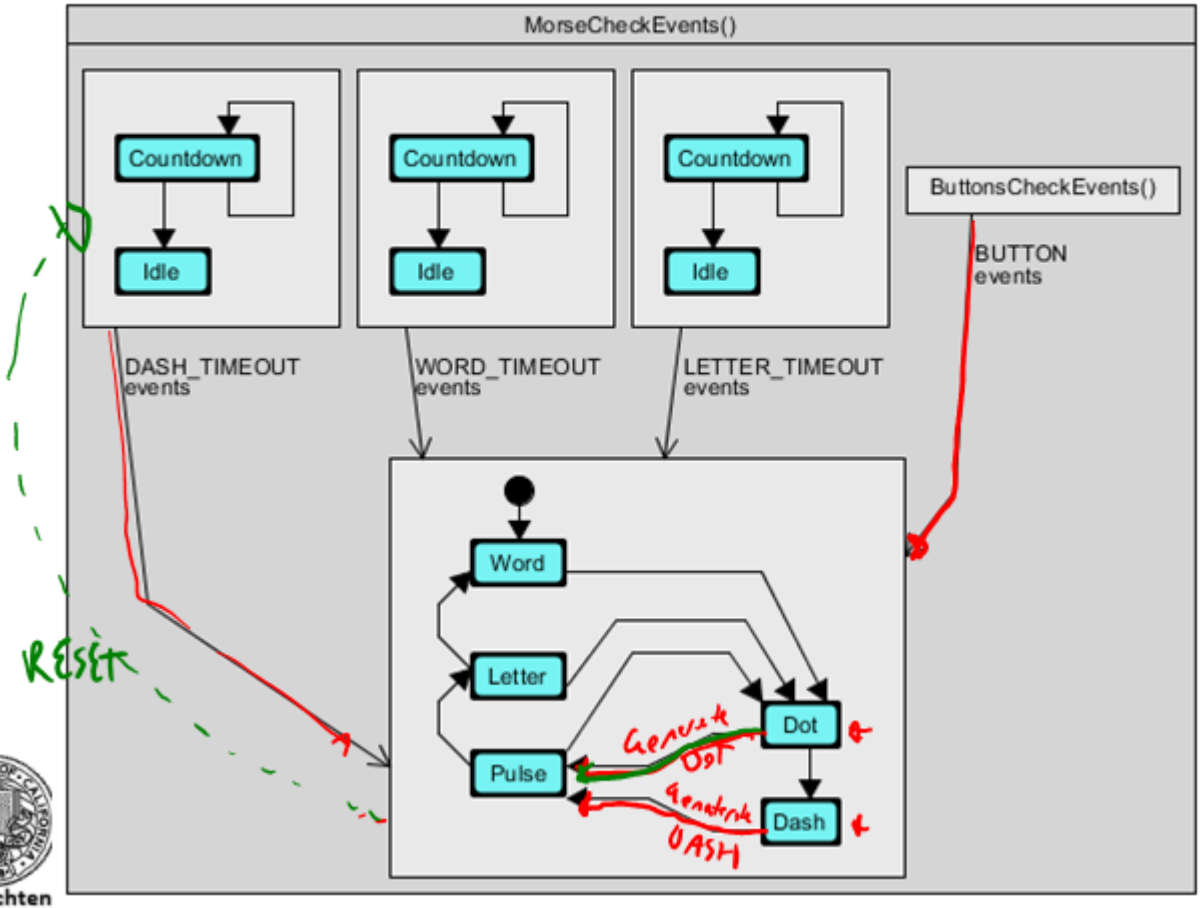
NEW_WORD

DASH

NEW_LETTER

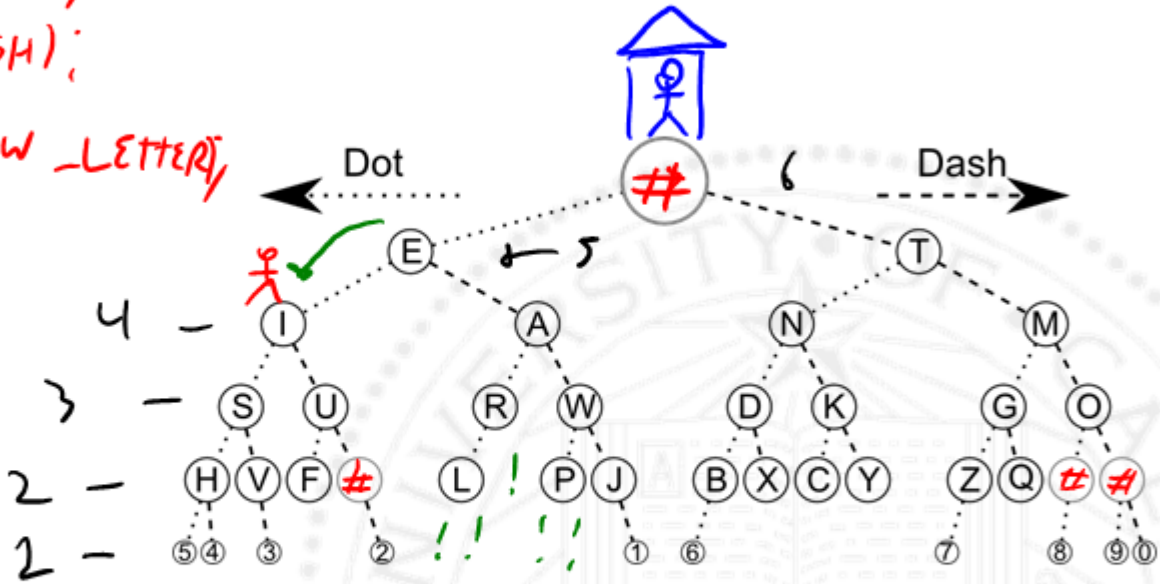


Inside MorseCheckEvents()

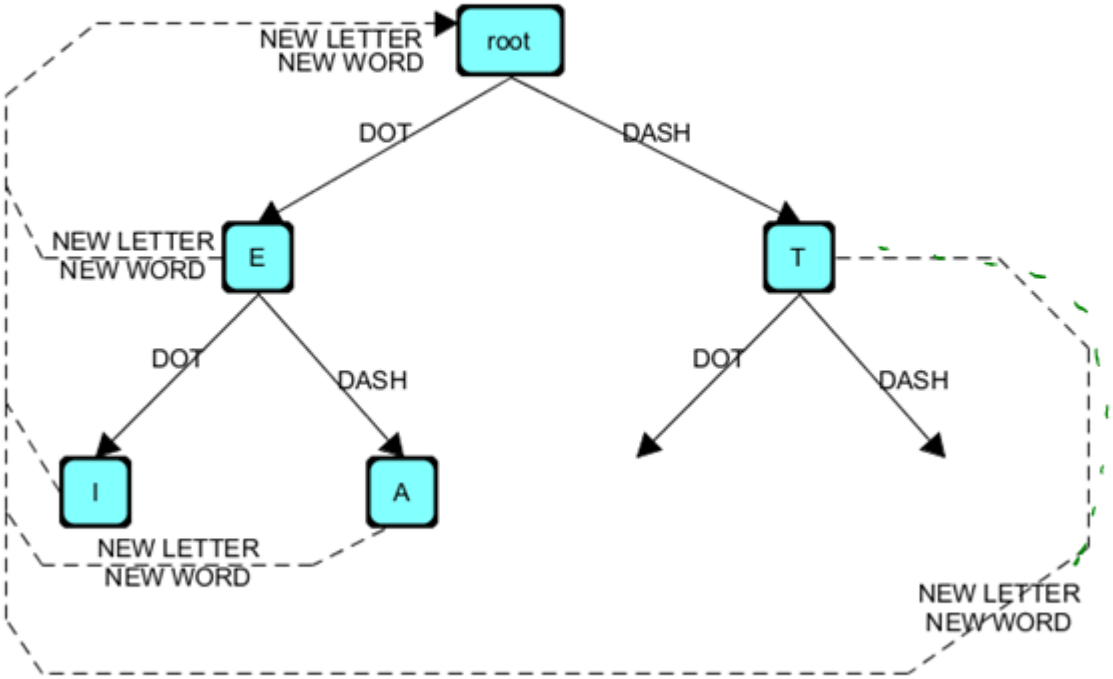


MD(DOT);
MD(DASH);
MD(NEW_LETTER);
LE'A'

MorseDecode()



MorseDecode() as SM?



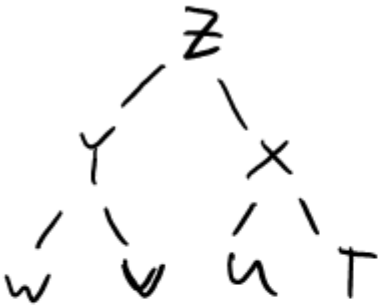
Events with Parameters:

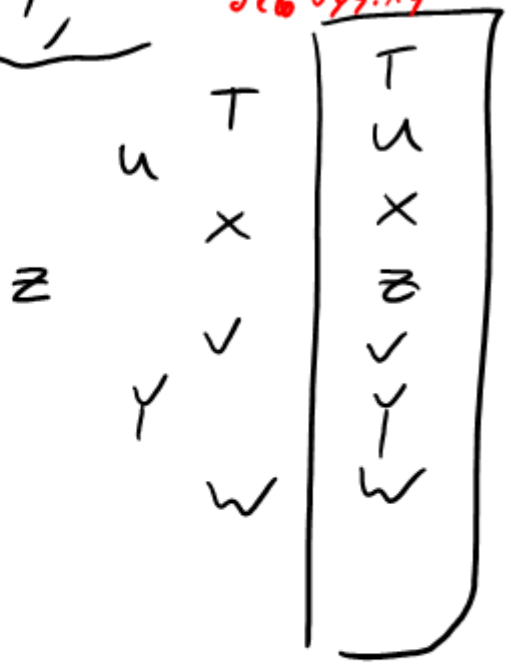
```
typedef enum {
    MORSE_EVENT_NONE,
    MORSE_EVENT_DOT,
    MORSE_EVENT_DASH,
    MORSE_EVENT_NEW_LETTER,
    MORSE_EVENT_NEW_WORD,
    MORSE_EVENT_CHAR_DECODED,
    MORSE_EVENT_ERROR,
} MorseEventType;
```

```
typedef struct{
    MorseEventType type;
    char parameter;
} MorseEvent;
```

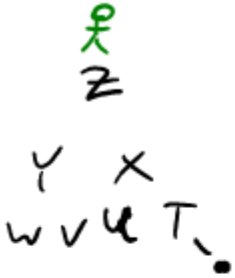
Tree Print();

Just for testing / debugging





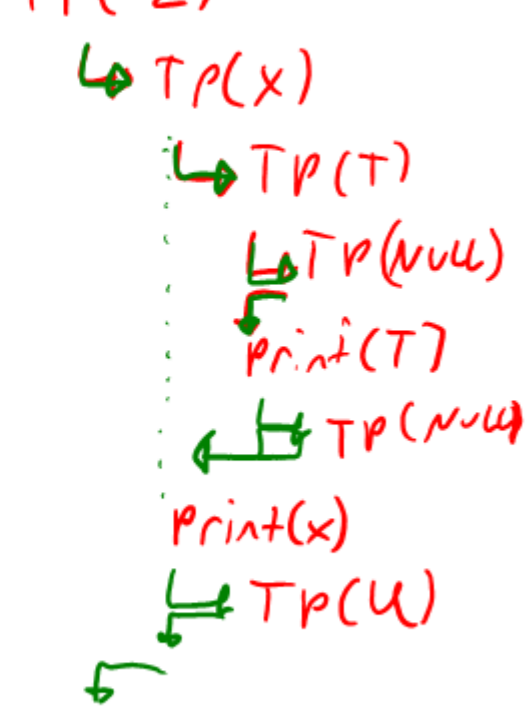
Tree Print:



T
X
u
Z

```
pseudocode(Node* self):
    if(      ==null)
        return;
    Tree Print (self->RC)
    print (self->data);

    Tree Print (self->LC)
    return
```



Tree create (*data, level)

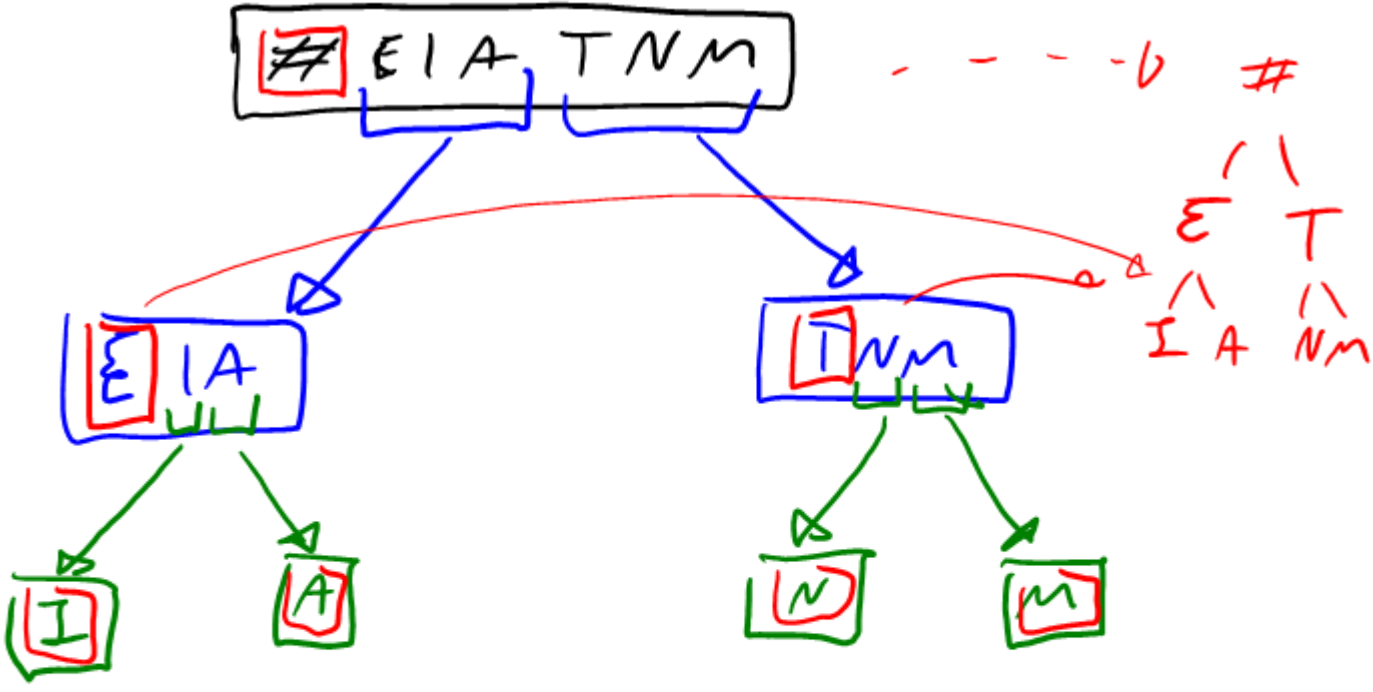
Tree create ("#EIA TNM", 3)

self data

left subtree's data

right subtree's data

```
TC (*data, level)
  if level = 0
      return
  malloc for self
  right child = Treecrate("TNM", level-1),
  left child = TC("EIA", level-1)
```



Morse Init():

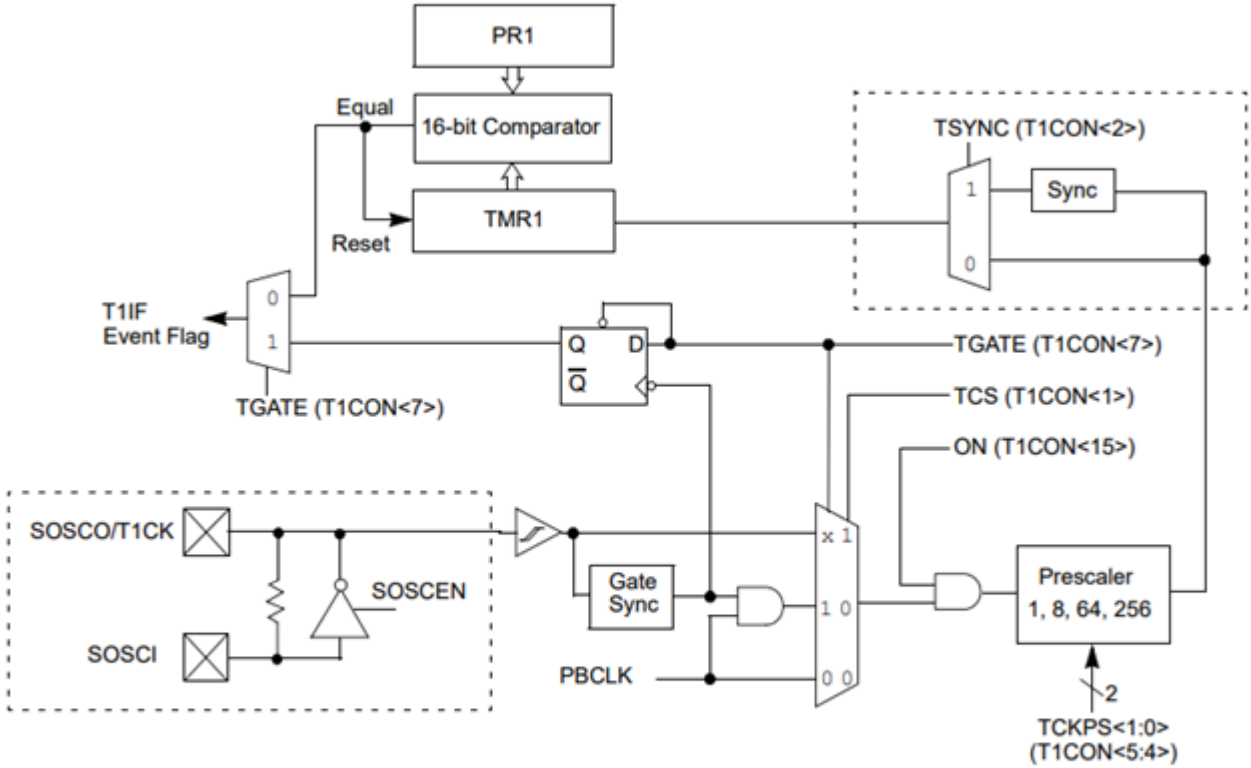
TreeCreate("

)

Other Lab 8 stuff?

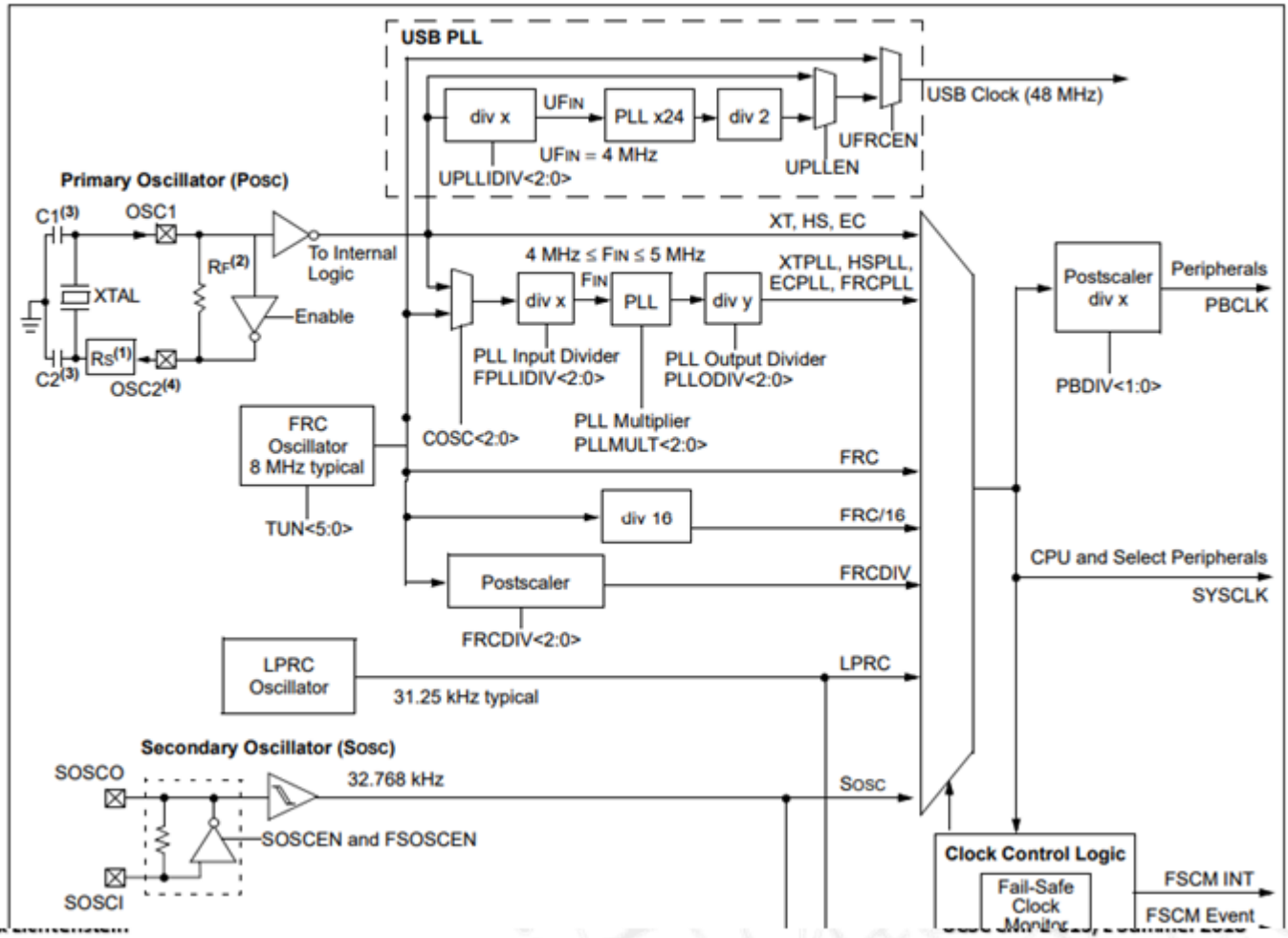
How fast can an ISR go?

FIGURE 13-1: TIMER1 BLOCK DIAGRAM(1)



Note 1: The default state of the SOSCEN (OSCCON<1>) during a device Reset is controlled by the FSOSCEN bit in Configuration Word DEVCFG1.

FIGURE 8-1: PIC32MX3XX/4XX FAMILY CLOCK DIAGRAM



bit 18-16 **PLLMULT<2:0>:** Phase-Locked Loop (PLL) Multiplier bits

The POR default is set by the FPLLMUL<2:0> bits (DEVCFG2<6:4>). Do not change these bits if the PLL is enabled. Refer to the **"Special Features"** chapter in the specific device data sheet for details.

111 = Clock is multiplied by 24
110 = Clock is multiplied by 21
101 = Clock is multiplied by 20
100 = Clock is multiplied by 19
011 = Clock is multiplied by 18
010 = Clock is multiplied by 17
001 = Clock is multiplied by 16
000 = Clock is multiplied by 15

High-Performance 32-bit RISC CPU:

- MIPS32® M4K® 32-bit core with 5-stage pipeline
- 80 MHz maximum frequency
- 1.56 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state Flash access
- Single-cycle multiply and high-performance divide unit
- MIPS16e® mode for up to 40% smaller code size
- Two sets of 32 core register files (32-bit) to reduce interrupt latency
- Prefetch Cache module to speed execution from Flash

```
void __ISR(_TIMER_1_VECTOR, IPL4AUTO) Timer1Handler(void) {
    // Clear the interrupt flag.
    INTClearFlag(INT_T1);

    // If we've exceeded the timer trigger count, trigger a timer event.
    if (++timerData.value > SWITCH_STATES()) {
        timerData.event = true;
        timerData.value = 0;
    }
}
```

What happens when an interrupt occurs?

```
!void __ISR(_TIMER_1_VECTOR, IPL4AUTO) Timer1Handler(void) {
0x9D0026F8: RDPGPR SP, SP
0x9D0026FC: MFC0 K1, EPC
0x9D002700: MFC0 K0, SRSCtl
0x9D002704: ADDIU SP, SP, -120
0x9D002708: SW K1, 116(SP)
0x9D00270C: MFC0 K1, Status
0x9D002710: SW K0, 108(SP)
0x9D002714: SW K1, 112(SP)
0x9D002718: INS K1, ZERO, 1, 15
0x9D00271C: ORI K1, K1, 4096
0x9D002720: MTC0 K1, Status
0x9D002724: SW V1, 28(SP)
0x9D002728: SW V0, 24(SP)
0x9D00272C: LW V1, 108(SP)
0x9D002730: ANDI V1, V1, 15
0x9D002734: BNE V1, ZERO, 0x9D002780
0x9D002738: NOP
0x9D00273C: SW RA, 92(SP)
0x9D002740: SW S8, 88(SP)
0x9D002744: SW T9, 84(SP)
0x9D002748: SW T8, 80(SP)
0x9D00274C: SW T7, 76(SP)
0x9D002750: SW T6, 72(SP)
0x9D002754: SW T5, 68(SP)
0x9D002758: SW T4, 64(SP)
```

```
0x9D002774: SW A1, 36(SP)
0x9D002778: SW A0, 32(SP)
0x9D00277C: SW AT, 20(SP)
0x9D002780: NOP
0x9D002784: MFLO V0
0x9D002788: SW V0, 100(SP)
0x9D00278C: MFHI V1
0x9D002790: SW V1, 96(SP)
0x9D002794: ADDU S8, SP, ZERO
!    // Clear the interrupt flag.
!    INTClearFlag(INT_T1);
0x9D002798: ADDIU A0, ZERO, 8
0x9D00279C: JAL INTClearFlag
0x9D0027A0: NOP
```

Commit