Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – hoseong Lee(이호성)

hslee00001@naver.com

자료구조

1. tree_delect
2. avl tree
3. 문제

Tree(me)

```
#include <stdio.h>
#include <malloc.h>

#define EMPTY 0

typedef struct tree
{
int data;
struct tree *left;
struct tree *right;
}Queue;

Queue *heap()
{
  Queue *tmp;
  tmp= (Queue *)malloc(sizeof(Queue));
  tmp->left = EMPTY;
  tmp->right = EMPTY;
  return tmp;
}
```

```
void ENQUEUE(Queue **head,int data)
{
  if(*head==NULL)
  {
    *head = heap();
    (*head)->data = data;
    return;
  }
  else if(data<((*head)->data))
  {
    ENQUEUE(&(*head)->left,data);
```

```
  }
  else if(data>((*head)->data))
  {
    ENQUEUE(&(*head)->right,data);
  }
  else
  return;
}

int print_queue(Queue *head)
{
       if(head)
       {
              printf("data=%d\n",head->data);
              print_queue(head->left);
              print_queue(head->right);
       }
}
```

```
void DEQUEUE(Queue *head,int data)
{
     Queue *tmp = head;

     if(head->data==data)
     {
        return 0;
     }
     else if(head→data>data)
    {
      head->left = DEQUEUE(head->left,data);
     }
     else if(head->data<data)
      head->right = DEQUEUE(head->right,data);
     }
```

```
    else
    {
      printf("%d 삭제한다잉",data);
      free(tmp);
      return head->
    }
}

int main(void)
{
Queue *head= EMPTY;
int arr[13]={50,45,73,32,48,46,16,37,120,47,130,127,124};
int i;
for(i=0;i<13;i++)
{
ENQUEUE(&head,arr[i]);
}
print_queue(head);
DEQUEUE(head,arr[0]);
return 0;

}
```

→ delect 부분이 아직도 안됨. 스택과 큐 다시보는중입니다

lec

```
#include <stdio.h>
#include <stdlib.h>

typedef struct __tree
{
	int data;
	struct __tree *left;
	struct __tree *right;
} tree;

tree *get_node(void)
{
	tree *tmp;
	tmp = (tree *)malloc(sizeof(tree));
	tmp->left = NULL;
	tmp->right = NULL;
	return tmp;
}
```

```
void tree_ins(tree **root, int data)  // 트리에 데이터 삽입
{
	if(*root == NULL)
	{
		*root = get_node();
		(*root)->data = data;
		return;
	}
	else if((*root)->data > data)
		tree_ins(&(*root)->left, data);
	else if((*root)->data < data)
		tree_ins(&(*root)->right, data);
}
```

```
void print_tree(tree *root) // 재귀함수로 트리 내용 프린
{
	if(root)
	{
		printf("data = %d, ", root->data);

		if(root->left)
			printf("left = %d, ", root->left->data);
		else
			printf("left = NULL, ");

		if(root->right)
			printf("right = %d\n", root->right->data);
		else
			printf("right = NULL\n");

		print_tree(root->left);
		print_tree(root->right);
	}
}
```

```
tree *chg_node(tree *root) // 데이터찾고, 동적할당 해제
{
	tree *tmp = root;

	if(!root->right)
		root = root->left;
	else if(!root->left)
		root = root->right;

	free(tmp);

	return root;
}
```

```
tree *find_max(tree *root, int *data)
{
	if(root->right)
		root->right = find_max(root->right, data);		// max 값은 right 이므로
	else
	{
		*data = root->data;
		root = chg_node(root);
	}

	return root;
}
```

```
tree *delete_tree(tree *root, int data)
{
	int num;
	tree *tmp;
	if(root == NULL)
	{
		printf("Not Found\n");
		return NULL;
	}
	else if(root->data > data)
		root->left = delete_tree(root->left, data);
	else if(root->data < data)
		root->right = delete_tree(root->right, data);
	else if(root->left && root->right)
	{
		root->left = find_max(root->left, &num);		// left, right 둘다 0 이 아니면  left 한칸  max 를 찾음.
		root->data = num;					//  max data
	}
	else
		root = chg_node(root);
	return root;
```

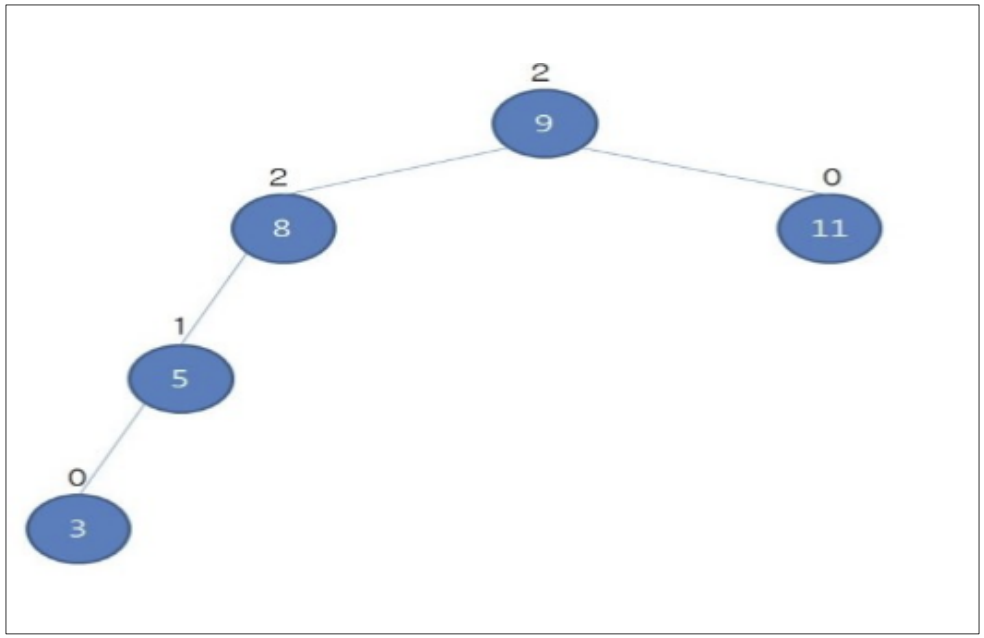
```
}
```

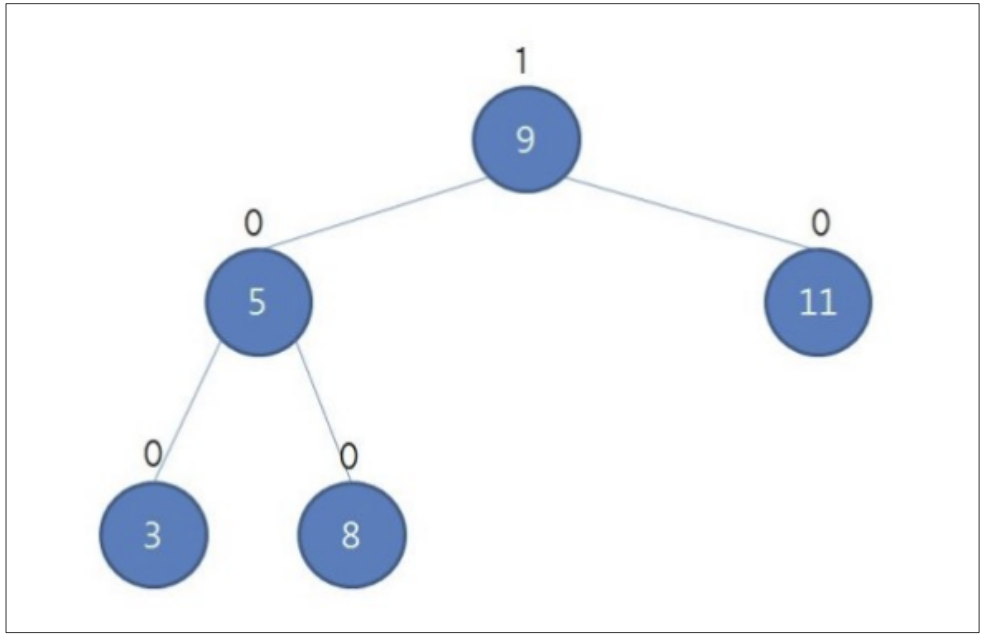
```
int main(void)
{
        int i;
        int data[14] = {50, 45, 73, 32, 48, 46, 16,
                37, 120, 47, 130, 127, 124};

        tree *root = NULL;

        for(i = 0; data[i]; i++)
                tree_ins(&root, data[i]);

        print_tree(root);

        delete_tree(root, 50);
        printf("After Delete\n");

        print_tree(root);

        return 0;
}
```

AVL 트리

이진트리의 찾는 효율을 올리기위해 만듬.

모든 노드의 서브트리 높이 차이가 1 이하라는 점이다. 만약 높이 차이가 2 이상이 된다면 노드들은 스스로 재배치하여 균형 상태를 유지한다. (=1 의 차이가 나게 만든다)





Quiz

1. 단 한 번의 연산으로 대소문자 전환을 할 수 있는 연산에 대해 기술하시오.

```
#include <stdio.h>

int main(void)
{
	char ASCII;
	scanf("%c",&ASCII);

	if(ASCII>= 'A' && ASCII<'Z')
		printf("%c",ASCII+32);			// 소문자로
	else
		printf("%c",ASCII-32);			 // 대문자로
	return 0;
}
```

2. Stack 및 Queue 외에 Tree 라는 자료구조가 있다. 이 중에서 Tree 는 Stack 이나 Queue 와는 다르게 어떠한 이점이 있는가 ?

스택과 queue 는 검색할때 다시 빼고 쌓고 해야하지만 트리구조로는 더 빠른 검색과 삽입이 용이하다.

3. 임의의 값 x 가 있는데, 이를 4096 단위로 정렬하고 싶다면 어떻게 해야할까 ? C 언어로 프로그래밍 하시오.

```
#include <stdio.h>

int main(void)
{
	int a,b,x;
	scanf("%d",&x);

	a=x%4096;
	b=x/4096;

	printf("%d + %d\n",a,b*4096);
}
```

4. int p[7] 와 int (*p)[7] 가 있다. 이 둘의 차이점에 대해 기술하시오.

int p[7] : int 형 데이터가 7 개를 저장할수있는 배열 p
int (*p)[7] : int 형 데이터를 7 개 저장할수있는 배열의 포인터 =(int (*)[7])

5. 다음을 C 언어로 구현하시오.
이번 문제의 힌트를 제공하자면 함수 포인터와 관련된 문제다.

아래와 같은 행렬을 생각해보자!
1 2 3
1 2 3
sapply(arr, func) 이라는 함수를 만들어서 위의 행렬을 아래와 같이 바꿔보자!
sapply 에 func 함수가 연산을 수행하는 함수로 만들어야 한다.
1 2 3
1 4 9

6. 다음 사항에 대해 아는대로 기술하시오. Intel Architecture 와 ARM Architecture 의 차이점은 무엇인가 ?

7. 이것이 없으면 C 언어의 함수를 호출할 수 없다. 여기서 이야기하는 이것은 무엇일까 ?

독립적 수행가능한 main() 함수를 가져야함.

8. 아래 내용을 C 로 구현해보도록 하자.
3x^2 + 7x 를 1 ~ 2 까지 정적분하는 프로그램을 구현해보라.

3x^2 에서 ^2 는 제곱을 의미한다.
예로 x 에 1 이 들어가면 3x^2 = 9 가 된다.

9. Memory Hierarchy(메모리 계층 구조)에 대해 기술하시오.
메인메모리, 레지스터, 캐시, 하드디스크 구조이다. 보조 기억장치는 주기억장치보다 늦고, 주기억장치는 캐시메모리보다 속도가 늦다.
주기억장치의 속도를 맞추기위해 메모리를 계층적으로 사용한다. 처리속도는: 래지스터 > 캐시메모리> 주기억장치> 보조기억장치

10. C 언어에서 중요시하는 메모리 구조에 대해 기술하시오.
(힌트: Stack, Heap, Data, Text 에 대해 기술하시오.)
메모리공간으로 text 영역, 데이터영역, 힙영역, 스택영역이 있다. text 영역은 코드가 저장되는 메모리 공간이다. 데이터영역은 전역변수와 static 변수가 할당되고, 프로그램 종료때까지 남아있다. 스택영역은 지역변수와 매개변수가 할당된다. 선언된 함수를 빠져나가면 없어진다.
heap 영역은 프로그래머가 원하는 시점에 변수를 할당하고 해제할 필요가있을 때 사용하는 영역이다.

11. 이프라인이 깨지는 경우에 대해 기술하시오.
분기가 발생할 때, 명령어들이 취소되면서 파이프라인이 깨지게된다. 분기 명령어는 파이프라인을 때려부순다. jmp 나 call 등의 분기명령어가 많을수록 힘들다.

12. void (* signal(int signum, void (* handler)(int)))(int)라는 signal 함수의 프로토타입을 기술하시오.
프로토타입을 기술하라는 의미는 반환형(리턴 타입)과 함수의 이름, 그리고 인자(파라메터)가 무엇인지 기술하라는 뜻임.

void(*)(int) signal(int signum, void(*handler)(int)

반환형(리턴 타입): int 형을 인자로 갖는 함수 포인터

함수 이름: signal

인자: int 형 signum 와 리턴이 없고, int 형을 인자로 갖는 함수 포인터 handler

13. goto 를 사용하는 이유에 대해 기술하시오.

jmp 를 사용할 때 파이프라인에 손실을 줌으로 if 문과 flag 문 사용을 안하므로 가속성이 좋다.

14. TI Cortex-R5F Safety MCU is very good to Real-Time System.
위의 문장에서 Safety MCU 가 몇 번째 부터 시작하는지 찾아내보자!

(배열의 인덱스로 표기해도 상관없고, 실제 위치로 표기해도 상관없으며 둘 중 무엇인지 표기하시오)

15. 배열에 아래와 같은 정보들이 들어있다.
2400, 5000, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 234, 345, 26023, 346, 345, 234, 457, 3, 1224, 34, 646, 732, 5,

4467, 45, 623, 4, 356, 45, 6, 123, 3245, 6567, 234, 567, 6789, 123, 2334, 345, 4576, 678, 789, 1000, 978, 456, 234756 , 234 ,4564 ,3243, 876,645, 534, 423, 312, 756, 235 ,75678
여기서 가장 빈도수가 높은 3 개의 숫자를 찾아 출력하시오!

16 비트 ADC 를 가지고 있는 장치가 있다.

보드는 12 V 로 동작하고 있고 ADC 는 -5 ~ 5 V 로 동작한다.

ADC 에서 읽은 값이 12677 일 때 이 신호를 전압으로 변환해보자!

(실제 보드가 없으므로 단순히 C 로 구현하면 된다)

17. 24 비트 DAC 장치가 있다.
이 장치는 -12 V ~ 12 V 로 동작하며 보드는 5 V 로 동작한다.

DAC 에서 나온 전압이 9.7 V 일 때 어떤 디지털 신호를 입력 받은것인지 파악해보자!

18. 운영체제의 5 대 요소 5 가지를 적으시오.

19. Stack 자료구조를 아래와 같은 포맷에 맞춰 구현해보시오.
(힌트: 이중 포인터)
ex)

```
int main(void)
{
 stack *top = NULL;
 push(&top, 1);
 push(&top, 2);
 printf("data = %d\n", pop(&top));
}
```

20. Binary Tree 나 AVL Tree, Red-Black Tree 와 같이 Tree 계열의 자료구조를 재귀 호출 없이 구현하고자 한다.
이 경우 반드시 필요한 것은 무엇인가 ?

22. AVL 트리는 검색속도가 빠르기로 유명하다. Red-Black 트리도 검색 속도가 빠르지만 AVL 트리보다 느리다. 그런데 어째서 SNS 솔루션 등에서는 AVL 트리가 아닌 RED-Blac 트리를 사용할까?

avl 트리는 데이터양이 많은 경우, 수많은 사용자가 몰려 데이터를 넣고 빼고 반복한다면 서버가 터지기 쉽다. rb 트리는 검색은 avl 보다 느리지만 삽입과 삭제속도가 빠르므로 sns 솔루션 등에서 좋다.

85. 기본 for 문 활용 문제이다.
1~100 까지의 숫자중 홀수만 더해서 출력해보시오.

→ 재귀없이 풀어보기.

Printf → 스택을 사용해보기

bull stack ~ // 현재 스택이 비어있는지 없는지 확인.

git init
git checkout -b xeno038-master master
git pull https://github.com/xeno038/Homework.git master

git checkout master
git merge –no-ff xeno038-master
git push origin master