

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 : Innova Lee(이상훈)

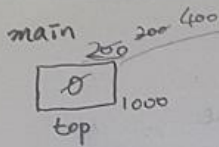
gcccompil3r@gmail.com

학생 : 황수정

sue100012@naver.com

8일차 (2018. 03. 05)

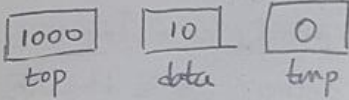
Stack 예시를 그림으로 표현 한 것.



Stack *top = EMPTY;

top 포인터가 가리키는 값
(top의 주소가 나타내는 값)
= 0

push



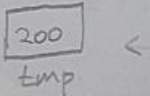
&top = **top int data;
top의 주소를 저장 (push(&top, 0)).

tmp = *top;

top 포인터 (top의 주소)가
가리키는 값을 tmp에 대입
현재 0이므로 tmp = 0

*top = get_node();

↳ get_node();

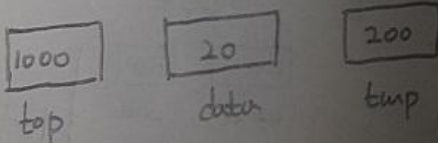


→ *top = get_node();

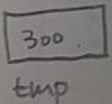
top 주소가 가리키는 값에
get_node() 함수 연산 결과를
대입하므로 '0' → 200이 된다.
(그리고 get_node()는 시작점/연산 종료)

⇒ 현재 push() 끝나므로 연산 종료로 push는 시작점

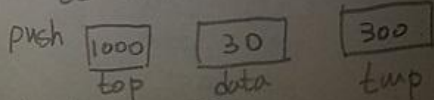
push.



get_node();



→ 같은 동작 실행



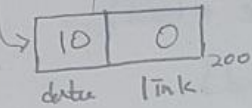
get node tmp 400

<heap>

tmp = (struct*) malloc (sizeof(struct));

→ 위의 struct node로

data 랑 link 생성



tmp → link = EMPTY;

초기화 선언.

get_node tmp로 link 접근.

return tmp;

↳ tmp를 반환하는데.

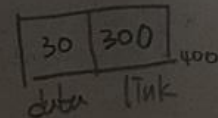
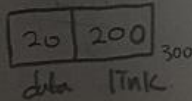
get_node malloc 된 곳의
주소를 반환해준다.

(*top) → data = data;

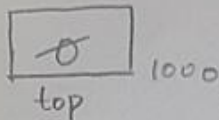
top 주소가 가리키는 data로
접근해서 data 값을 대입한다.

(*top) → link = tmp;

tmp 값은 0이니 0 입력.

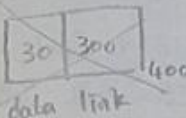
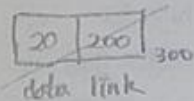
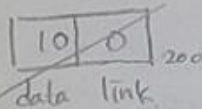


main 200 300 400 300 200 0

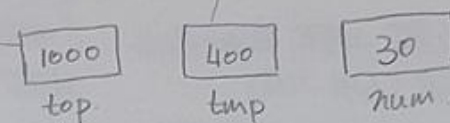


*top = (*top) -> link;
top 주소가 가리키는 곳의 link로
접근해 그 값을 top에 대입

heap



pop(&top)



&top = *top

top의 주소를 담는다

tmp = *top;

top 주소가 가리키는 값 대입

*top가 0이 아니므로 아무 문제

num = tmp -> data;

tmp가 가리키는 data에 접근해서

그 값을 대입하라 30이 된다.

→ 가 설정된 상태로

push가 끝났다

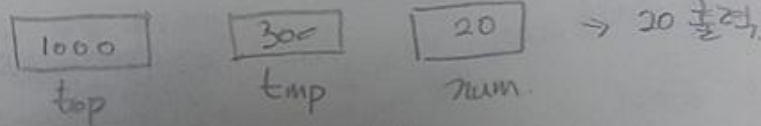
이제 pop()을 실행하면 된다.

free(tmp); → tmp가 가리키는 malloc 닫기

return num;

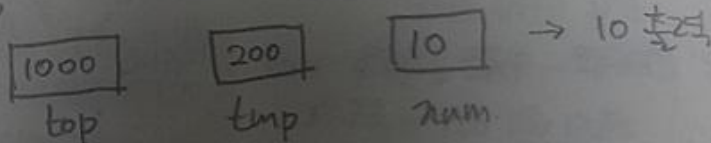
이제 30을 반환하고 pop 함수는 끝난다

pop



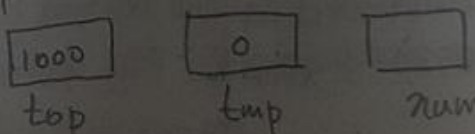
→ 위와 같은 동작 실행

pop



→ 위와 같은 동작 실행

pop



IF문 실행됨 → "Stack is empty!!" 출력

return 0; 이라

Stack is empty!! 후 0도 출력된다

data = 10, 20, 30 삽입한다.

Void enqueue (queue **head, int data)

```

{
    if (*head == NULL)
    {
        *head = get_node();
        (*head) -> data = data;
        return;
    }
    enqueue (&(*head) -> link, data);
    printf ("test \n");
}

```

Void print - queue (queue * head)

```

{
    queue *tmp = head;
    while (tmp)
    {
        printf ("%d \n", tmp -> data);
        tmp = tmp -> link;
    }
}

```

print - queue

1000 200
head tmp

*tmp = head
tmp 가 가리키는 곳에 head
값을 넣어라

while (tmp)

printf ("%d \n", tmp -> data);

→ tmp 가 가리키는 data로 접근

'10' 출력

→ tmp 가 가리키는 link 접근
tmp에 대입

→ 300 으로 data 접근

'20' 출력

→ tmp = tmp -> link로

400 대입

→ tmp -> data로 '30' 출력

main

200 1000
head

enqueue

1000 10
head data

*head
이런 포인터
head 주소가
→ *head = 0 이므로 아무것도

get node

200 tmp

→ *head = get_node();
이므로 0 → 200 이된다.

(*head) -> data = data;

head 주소가 가리키는 data에

접근해서 data 값을 대입하라.

return 으로 끝이 나지만

재귀 호출이라 stack 의 push()

pop() 과 다르게 값이 남아있다.

*자리가 부족하여.

용건

enqueue

1000 20
head data

→ *head = 0 이아니므로

enqueue

204 20
head data

&(*head) -> link

head 가 가리키는 곳 link
로 접근해 주소를 알아라

if 으로 다시 반복하면

if 충족하므로

get node

300 tmp

400 300
data link

enqueue

1000 30
head data

enqueue

204 30
head data

enqueue

304 30
head data

get node 400 tmp

30 0 40
data link

과제2. c언어 복습 문제

1~1000 사이에 4나 6으로 나누어도 나머지가 1인 수의 합을 구하라.

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

```
int syn(int start, int end, int t1, int t2)
{
    int res = 0, i = start;

    while(i < end + 1)
    {
        if(((i % 4) == 1) || ((i % 6) == 1))
        {
            res += i;
        }
        i++;
    }

    return res;
}
```

```
int main(void)
{
    printf("tot series sum = %d\n", syn(1, 1000, 4, 6));
    return 0;
}
```

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

```
int sum(int s, int e, int t1, int t2)
{
    int i, sum = 0;

    for(i = s; i <= e; i++)
        if(((i % 4) == 1) || ((i % 6) == 1))
            sum += i;

    return sum;
}

int main(void)
{
    printf("sum of 3 series = %d\n", sum(1, 1000, 4, 6));
    return 0;
}
```

위의 두 개의 코드는 선생님께서 각각 While과 for로 잡으신 코드이다.
< 처음 문제 풀이 당시 잡았던 코드이다. 합이 아닌 수의 값들이 출력 되었다.

```
#include<stdio.h>
int main(void)
{
    int a = 1;
    while (a <= 1000)
    {
        if (a == 1)
        {
            a++;
        }
        else
        {
            if (a % 4 == 1)
            if (a % 6 == 1)
                printf("%d\n", a);
            a++;
        }
    }
}
```

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

```
int sum(int a, int b)
{
    int i, sum = 0;
    for (i = a; i <= b; i++)
        if (((i % 4) == 1) || ((i % 6) == 1))
            sum += i;
    return sum;
}

int main(void)
{
    printf("1~1000 사이에 4나 6으로 나누어도
    나머지가 1인 수의 합은 %d이다\n", sum(1, 1000));
    return 0;
}
```

<엮은 다시 코드를
잡아 본 것으로
합이 도출되어
166167 이 나왔다.