EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv1 과정

제 4기

2022. 10. 27

진동민

목차

EDDI
Electronic Design
Development Institute

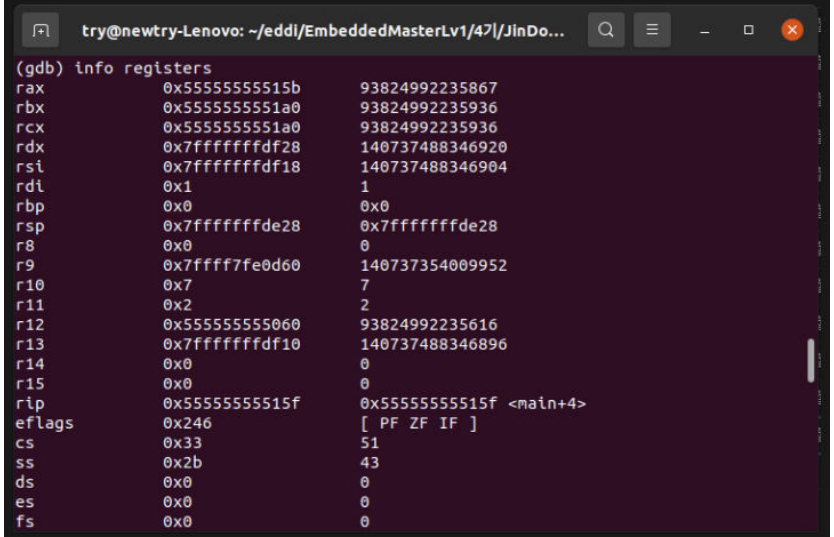
질문은 날짜 순서대로 기입

gdb에서의 레지스터 관련 명령어 질문

EDDI
Electronic Design Development Institute

Reason

1. gdb에서 'info registers' 명령어를 실행했을 때 가운데와 오른쪽의 숫자는 어떤 것을 의미하는건가요?



2. 어떤 레지스터 값을 들여다보고 싶을 때 쓰는 명령어인 'x $레지스터 변수' 를 썼을 때, 왼쪽과 오른쪽은 어떤 것을 의미하나요?

```
(gdb) x $rsp
0x7fffffffde20: 0x00000000
```

Solution

레지스터 내부의 값을 hex값으로 보여주는 것이 중간 부분,

가장 오른쪽은 10진수 값을 보여줍니다.

어셈블리어 개발 질문

EDDI
Electronic Design Development Institute

Reason

어셈블리어로 개발하게 된다면 스택 관리까지 생각(고려 및 계산)하면서 개발을 해야하나요?

Try

Solution

네 지역변수가 스택에 배치되고 함수 파라메터가 스택에 배치되기 때문에 고려해야합니다.

파이썬 설계 질문

EDDI
Electronic Design Development Institute

Reason

학교에서 한 과목의 평가방법으로 파이썬 tkinter 모듈을 사용하여 데스크탑 프로그램을 개발하는 프로젝트를 하게 되었습니다.

카페에서 올라오는 글을 보면 강사님께서 유연한 설계가 중요하다라고 말씀하신 것을 종종 보는데요.

간단한 문법을 장점으로 지닌 파이썬으로 프로젝트를 개발하게되더라도 지금 생각나는 것은 파일분할 밖에 생각이 나지 않는데, 개발하면서 유연한 설계를 위해 고려해야할 점이라던가 상황이 있을까요?

Try

Solution

파이썬으로 설계한다면 클래스를 통한 OOP,

그리고 의존성들을 분리하는 작업입니다.

오픈소스 참여 시 이점

EDDI
Electronic Design Development Institute

Reason

GitHub에서 C 언어 오픈소스 프로젝트에 참여하면 회사 이직할 때 커리어로 인정이 되나요?

한국은 어떤 추세인지 궁금합니다.

Try

Solution

소프트웨어 업종에서는 많은 인정을 해주는 편입니다.

커리어 측면에서도 나쁠것이 없습니다.

임베디드에 트렌드는 무엇인가

EDDI
Electronic Design Development Institute

Reason

카페 게시글(https://cafe.naver.com/eddicorp/356)에서 웹 UI/UX의 경우에는 React, Vue, Svelte 3대장이라 불릴 정도로 트렌드로 자리잡고있는데요.

임베디드에도 트렌드라고 하는 무언가가 있을까요?

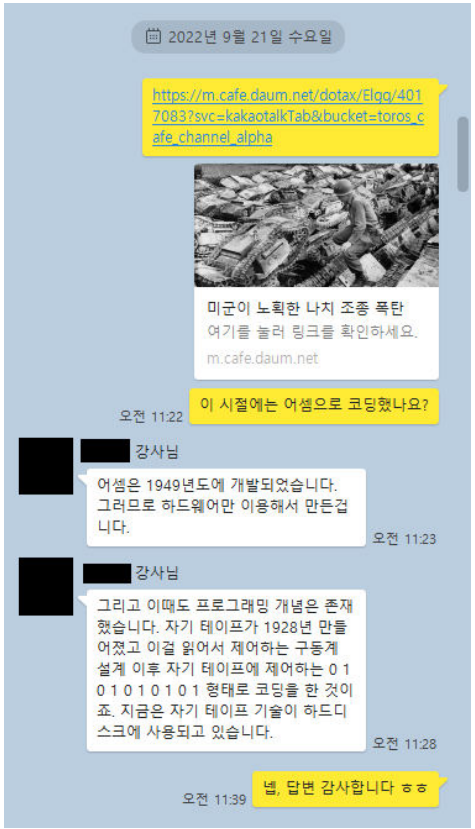
Try

Solution

IoT, 자율주행, 인공지능(엣지 디바이스)로 볼 수 있겠습니다.

결론적으로 전부 다 융합 분야에 해당합니다.

나치가 만든 원격조종 자폭무기는 어셈으로 코딩했는가?

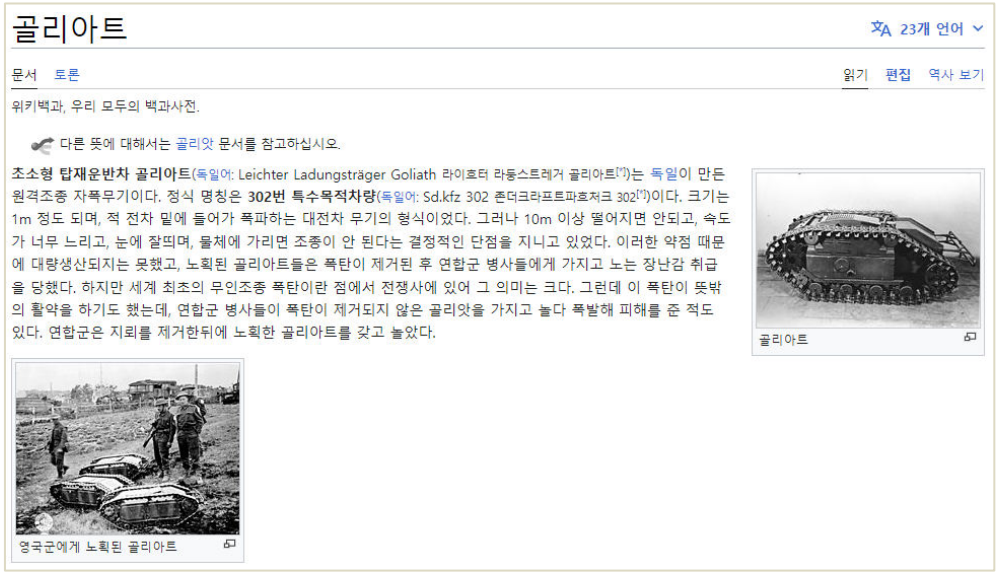


해당 링크:
https://m.cafe.daum.net/dotax/Elgq/4017083?svc=kakaotalkTab&bucket=toros_cafe_channel_alpha

카카오톡에서 보여주는 컨텐츠 중에서 나치가 만든 원격 조종 자폭무기 글을 보았다.

이 시절에도 어셈블리어로 개발을 했는지 궁금해서 물어보았는데, 바로 답변을 해주셨다 ㅎㄷㄷ

나치가 만든 원격조종 자폭무기는 어셈으로 코딩했는가



골리아트

23개 언어

문서 토론 읽기 편집 역사 보기

위키백과, 우리 모두의 백과사전.

다른 뜻에 대해서는 골리앗 문서를 참고하십시오.

초소형 탑재운반차 골리아트(독일어: Leichter Ladungsträger Goliath 라이히터 라둥스트레거 골리아트[1])는 독일이 만든 원격조종 자폭무기이다. 정식 명칭은 **302번 특수목적차량**(독일어: Sd.kfz 302 존더크라프트파흐처크 302[1])이다. 크기는 1m 정도 되며, 적 전차 밑에 들어가 폭파하는 대전차 무기의 형식이었다. 그러나 10m 이상 떨어지면 안되고, 속도가 너무 느리고, 눈에 잘띄며, 물체에 가리면 조종이 안 된다는 결정적인 단점을 지니고 있었다. 이러한 약점 때문에 대량생산되지는 못했고, 노획된 골리아트들은 폭탄이 제거된 후 연합군 병사들에게 가지고 노는 장난감 취급을 당했다. 하지만 세계 최초의 무인조종 폭탄이란 점에서 전쟁사에 있어 그 의미는 크다. 그런데 이 폭탄이 뜻밖의 활약을 하기도 했는데, 연합군 병사들이 폭탄이 제거되지 않은 골리앗을 가지고 놀다 폭발해 피해를 준 적도 있다. 연합군은 지뢰를 제거한뒤에 노획한 골리아트를 갖고 놀았다.

위키피디아

추가로 구글에 검색을 해보니 '골리아트'라고 한다.

위키피디아에서는 **'세계 최초의 무인조종 폭탄이란 점에서 전쟁사에 있어 그 의미는 크다.'** 라고 적혀있는데 역시 기술이 중요한 것 같다…

인공지능 기초 책 추천

EDDI
Electronic Design Development Institute

Reason

(어느 주제에 작성할지 고민하다가, 인공지능이 파이썬하고 밀접하게 관련있는 것 같아 파이썬 이슈 사항에 작성하게되었습니다)

인공지능의 완전 기본부터 배울 수 있는 책 추천해주실 수 있으실까요?

인공지능 기초 책 추천

EDDI
Electronic Design Development Institute

Solution

답변을 작성하는데 꽤 시간이 걸릴것 같네요.

우선 인공지능이라는 분야를 제대로 학습하기 위해서는 수학 공부가 우선되어야 합니다.

사실 인공지능을 기초부터 배울 수 있는 책이 있다고 누군가가 얘기한다면

그것은 어떻게 보면 반쯤은 사기라고 볼 수도 있습니다.

제대로 공부하기 위해서는 아래와 같은 기초들이 필요하기 때문입니다.

(머리속에서 생각한 내용들을 프로그래밍 할 수 있다는 것을 기본 전재로 깔았습니다)

1. 대학수학
   당장 아래에서 학습할 공업수학 내용을 파악하기 위해 기초 토대가 됩니다.

2. 공업수학
   인공지능이라는 것이 내부 메커니즘들은
   내부적으로 신호 처리 메커니즘에 기반하여 동작하고 있습니다.
   이것의 기초 토대는 라플라스 변환과 푸리에 변환입니다.
   또한 여기서도 여전히 차분 개념이 사용되며 차분 방정식을 풀기도 합니다.
   이와 같은 기초 토대를 쌓기 위해 학습해야 하는 사항들입니다.

인공지능 기초 책 추천

EDDI
Electronic Design Development Institute

3. 디지털 신호 처리
   컴퓨터로 다뤄지는 모든 데이터는 연속적인 아날로그 신호가 아닌
   불연속적인 이산 데이터로 다뤄집니다.
   이 이산 데이터를 샘플링 데이터라고 하며
   샘플 값을 가지고 데이터를 해석하는 것을 디지털 신호 처리라고하죠.
   실제로 이때 위의 1번, 2번 사항을 프로그래밍으로 구현할 수 있는 수준까지 되면 좋습니다.

4. 통계학
   인공지능 분야는 통계학을 응용하는 분야와 응용하지 않는 분야가 나뉩니다.
   기본적인 통계 (확률 분포, 기대값, 밀도 함수, Z Score, 베이지안 추론 등등)은 알고 있어야 합니다.

5. 인공지능
   이후에 인공지능 공부를 하면 책이 어느정도 읽혀집니다.
   부가적으로 TensorFlow나 Keras를 별도로 학습해야 합니다.
   (개인적으로 초보짬에선 Keras 추천합니다)
   추후 인공지능을 어떤 분야에 적용하냐에 따라
   각 분야별 추가적으로 공부해야 하는 별도의 과목들이 존재합니다.
   (기초라고 해서 딱 기초 수준에 맞게 답변 작성했습니다)

결론: 기초라고 하지만 이 녀석은 기본적으로 융합 분야이기 때문에
   어떤 책을 보고 인공지능이 무엇이다라고 섣불리 답하기 어렵다는 것입니다.
   (인공지능이 무엇인지 알려면 위의 5가지는 기본적으로 공부가 되어야 합니다)

소스코드 분석을 위한 GitHub 오픈소스 프로젝트 추천

EDDI
Electronic Design Development Institute

Reason

오픈소스에 관심이 많아서 가끔 여유있을 때 소스코드 분석을 하려고 합니다.

임베디드 마스터 레벨1 수준에서 C언어 오픈소스 프로젝트를 추천해주실 수 있으신가요?

Try

Solution

1. 리눅스 커널
2. FFMPEG

C로 아주 잘 만들어진 프로젝트중 Top 2 입니다.

배열의 어셈블리어 해석에 관한 질문

EDDI
Electronic Design Development Institute

Reason

```c
#include <stdio.h>

int main(void)
{
    // 배열의 선언 방법
    // 1. 데이터 타입 작성 (int, float, double 등등)
    // 2. 변수 선언하듯이 이름 작성
    // 3. 대괄호 열고 몇 개를 만들지 숫자를 작성
    // 4. 필요하다면 { } 중괄호 내부에 배치할 값들을 작성

    int array[5] = { 1, 2, 3, 4 };
    int i;

    for (i = 0; i < 5; i++)
    {
        printf("array[%d] = %d\n", i, array[i]);
    }

    return 0;
}
```

carbon
carbon.now.sh

array.c

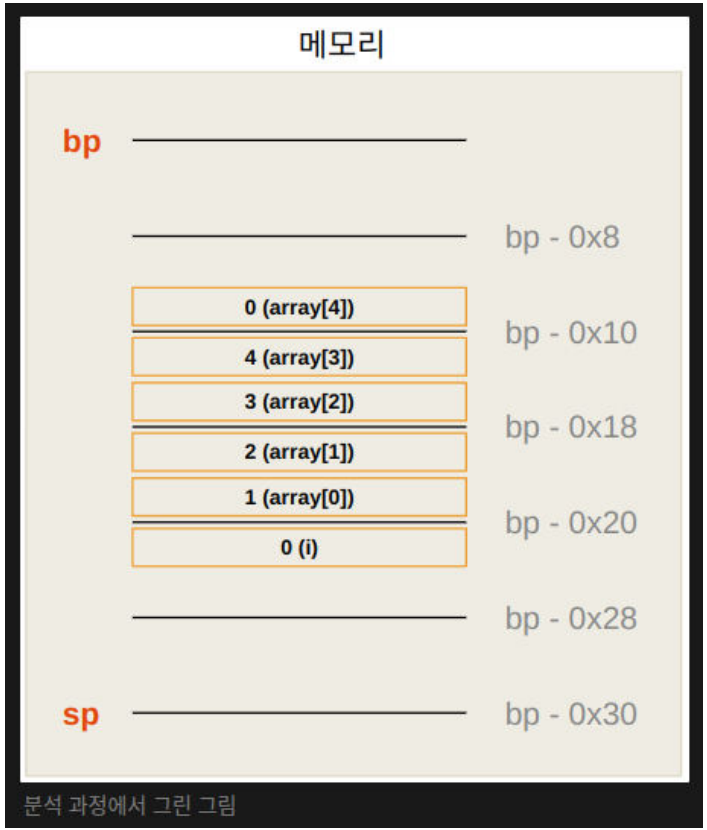
배열의 어셈블리어 해석에 관한 질문

EDDI
Electronic Design Development Institute

```
(gdb) disas
Dump of assembler code for function main:
=> 0x0000555555555169 <+0>:     endbr64
   0x000055555555516d <+4>:     push   %rbp
   0x000055555555516e <+5>:     mov    %rsp,%rbp
   0x0000555555555171 <+8>:     sub    $0x30,%rsp
   0x0000555555555175 <+12>:    mov    %fs:0x28,%rax
   0x000055555555517e <+21>:    mov    %rax,-0x8(%rbp)
   0x0000555555555182 <+25>:    xor    %eax,%eax
   0x0000555555555184 <+27>:    movq   $0x0,-0x20(%rbp)
   0x000055555555518c <+35>:    movq   $0x0,-0x18(%rbp)
   0x0000555555555194 <+43>:    movl   $0x0,-0x10(%rbp)
   0x000055555555519b <+50>:    movl   $0x1,-0x20(%rbp)
   0x00005555555551a2 <+57>:    movl   $0x2,-0x1c(%rbp)
   0x00005555555551a9 <+64>:    movl   $0x3,-0x18(%rbp)
   0x00005555555551b0 <+71>:    movl   $0x4,-0x14(%rbp)
   0x00005555555551b7 <+78>:    movl   $0x0,-0x24(%rbp)
   0x00005555555551be <+85>:    jmp    0x5555555551e3 <main+122>
   0x00005555555551c0 <+87>:    mov    -0x24(%rbp),%eax
   0x00005555555551c3 <+90>:    cltq
   0x00005555555551c5 <+92>:    mov    -0x20(%rbp,%rax,4),%edx
   0x00005555555551c9 <+96>:    mov    -0x24(%rbp),%eax
   0x00005555555551cc <+99>:    mov    %eax,%esi
   0x00005555555551ce <+101>:   lea    0xe2f(%rip),%rdi        # 0x555555556004
   0x00005555555551d5 <+108>:   mov    $0x0,%eax
   0x00005555555551da <+113>:   callq  0x555555555070 <printf@plt>
   0x00005555555551df <+118>:   addl   $0x1,-0x24(%rbp)
   0x00005555555551e3 <+122>:   cmpl   $0x4,-0x24(%rbp)
   0x00005555555551e7 <+126>:   jle    0x5555555551c0 <main+87>
   0x00005555555551e9 <+128>:   mov    $0x0,%eax
   0x00005555555551ee <+133>:   mov    -0x8(%rbp),%rcx
   0x00005555555551f2 <+137>:   xor    %fs:0x28,%rcx
   0x00005555555551fb <+146>:   je     0x555555555202 <main+153>
   0x00005555555551fd <+148>:   callq  0x555555555060 <__stack_chk_fail@plt>
   0x0000555555555202 <+153>:   leaveq
   0x0000555555555203 <+154>:   retq
End of assembler dump.
```

array.c의 어셈블리어

배열의 어셈블리어 해석에 관한 질문



배열의 어셈블리어 해석에 관한 질문

EDDI
Electronic Design Development Institute

이 중에서 +92번째 명령어의 의미를 알고 싶습니다.

```
mov -0x20(%rbp, %rax, 4), %edx
```

아마 제 추측으로는 for문을 이용하여 순서대로 배열 요소에 접근하는 것이니, (%rbp + %rax * -4) - 0x20 위치에 저장된 값을 edx로 복사하는 것이라 생각을 했습니다.

이게 제 생각이지만 정확한 해석을 알고 싶어서 질문했습니다.

배열의 어셈블리어 해석에 관한 질문

EDDI
Electronic Design Development Institute

Solution

rax는 i 값에 해당하므로 실제 루프를 돌면서 값이 하나씩 올라갈 것입니다.

또한 rbp - 0x20은 배열의 시작 주소죠.

시작 주소 부터 4바이트씩 전진하면서 후속 배열을 참조하는 것입니다.

결론적으로 rbp -0x20 + rax * 4)를 의미합니다.

rax 가 rbp - 0x24로 i 값임을 유의합시다!

의존성 분리는 어디서 하는가

EDDI
Electronic Design Development Institute

Reason

의존성 분리는 구현이 아닌 백로그를 작성할 때 고려해서 분리 되는 것인가요?

Try

Solution

의존성 분리라는 것은 코드 레벨에서 분리하는 것입니다.

재활용성을 높이기 위한 사항중 하나입니다.