

어셈블리어

Assembly Language

이다영

1. 어셈블리어란?



```
1000 1011 0100 0101 1111 1000  
1000 0011 1100 0100 0000 1100  
0000 0011 0100 0101 1111 1100
```

기계어



1000 1011

The diagram illustrates the assembly process of a machine instruction. It features a light gray rectangular background. On the left, a green circle contains the binary code '1000 1011'. A dark gray arrow points from this circle to a teal circle on the right, which contains the assembly instruction 'MOV'. Below the teal circle, the Korean text '어셈블리어' (Assembler) is written in a dark gray font.

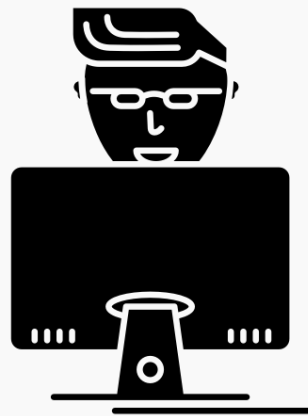
MOV

어셈블리어

저급언어 & 고급언어



저급언어



고급언어

기본 구조 & 문법

ADD Operand1, Operand2

Destination Source

Source Destination

AT & T

Intel

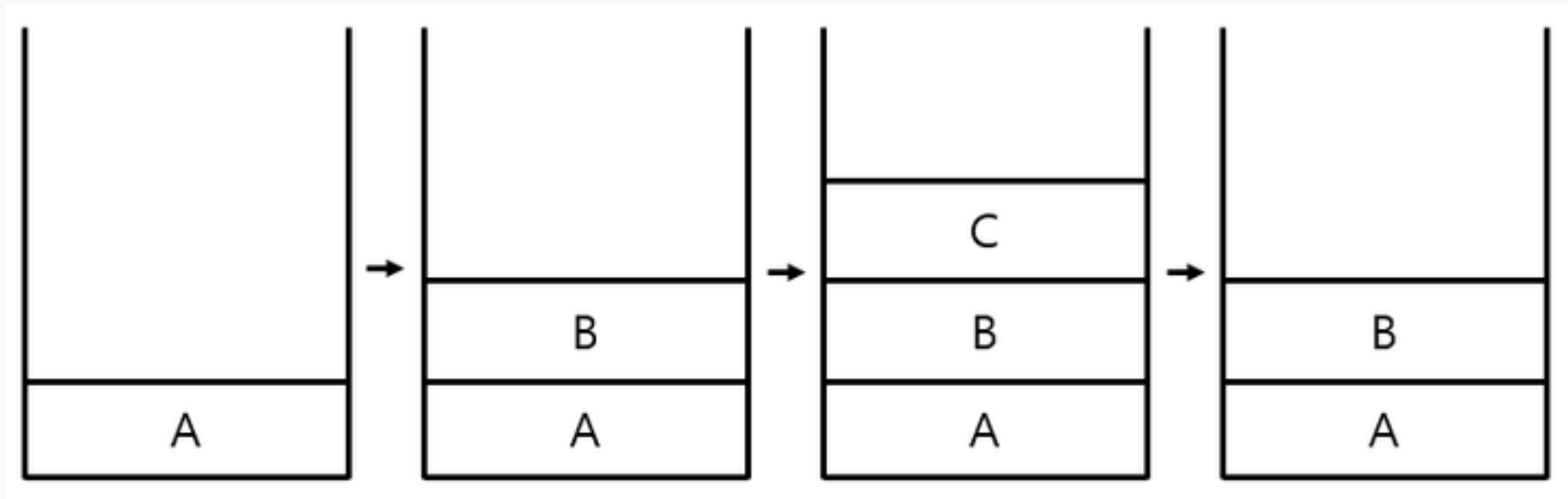
2. 레지스터

Register

: CPU 내부에서 데이터를 일시적으로 저장하는 장소

종류	레지스터	용도
범용	EAX	산술/논리 연산, 처리 결과 리턴값 저장
	EDX	산술/논리 연산 보조
포인터	ESP	현재 스택의 가장 위에 들어있는 데이터를 가리킴
	EBP	현재 스택의 가장 바닥을 가리킴

낮은 주소



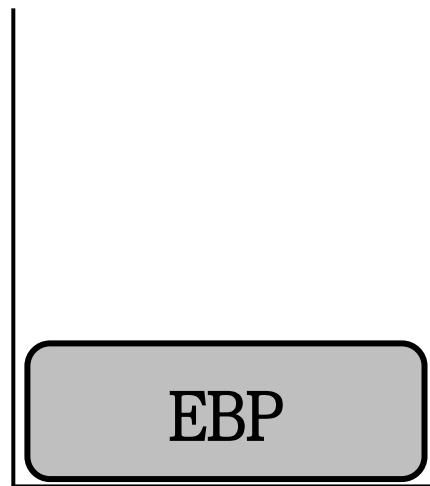
높은 주소

3. 핵심 명령어

Opcode

PUSH EBP

POP EBP



스택

주소의 값



MOV EAX, [EBP+8]

LEA EAX, [EBP+8]



주소

ADD

EAX = 50

ADD EAX, 10

$EAX = EAX + 10 = 60$

SUB

EAX = 50

SUB EAX, 10

$EAX = EAX - 10 = 40$

CALL

RET

```
1 #include <stdio.h>
2
3 int Sum()
4 {
5     int a = 1;
6     int b = 2;
7
8     return a + b;
9 }
10
11 int main()
12 {
13     int a;
14     a = Sum();
15
16     return 0;
17 }
18
```

4. 분석

```

int main()
{
int a = 10, b = 20, c;
c = a + b;
printf("%d", c);
return 0;
}

```

Dump of assembler code for function main:

0x0804841d	<+0>:	push	ebp
0x0804841e	<+1>:	mov	ebp,esp
0x08048420	<+3>:	sub	esp,0x14
0x08048423	<+6>:	mov	DWORD PTR [ebp-0x4],0xa
0x0804842a	<+13>:	mov	DWORD PTR [ebp-0x8],0x14
0x08048431	<+20>:	mov	DWORD PTR [ebp-0xc],0x0
0x08048438	<+27>:	mov	eax,DWORD PTR [ebp-0x8]
0x0804843b	<+30>:	mov	edx,DWORD PTR [ebp-0x4]
0x0804843e	<+33>:	add	eax,edx
0x08048440	<+35>:	mov	DWORD PTR [ebp-0xc],eax
0x08048443	<+38>:	mov	eax,DWORD PTR [ebp-0xc]
0x08048446	<+41>:	mov	DWORD PTR [esp+0x4],eax
0x0804844a	<+45>:	mov	DWORD PTR [esp],0x80484f0
0x08048451	<+52>:	call	0x80482f0 <printf@plt>
0x08048456	<+57>:	leave	
0x08048457	<+58>:	ret	

End of assembler dump.

Dump of assembler code for function main:

```
0x0804841d <+0>:  push ebp
0x0804841e <+1>:  mov  ebp,esp
0x08048420 <+3>:  sub  esp,0x14
0x08048423 <+6>:  mov  DWORD PTR [ebp-0x4],0xa
0x0804842a <+13>:  mov  DWORD PTR [ebp-0x8],0x14
0x08048431 <+20>:  mov  DWORD PTR [ebp-0xc],0x0
0x08048438 <+27>:  mov  eax,DWORD PTR [ebp-0x8]
0x0804843b <+30>:  mov  edx,DWORD PTR [ebp-0x4]
0x0804843e <+33>:  add  eax,edx
0x08048440 <+35>:  mov  DWORD PTR [ebp-0xc],eax
0x08048443 <+38>:  mov  eax,DWORD PTR [ebp-0xc]
0x08048446 <+41>:  mov  DWORD PTR [esp+0x4],eax
0x0804844a <+45>:  mov  DWORD PTR [esp],0x80484f0
0x08048451 <+52>:  call 0x80482f0 <printf@plt>
0x08048456 <+57>:  leave
0x08048457 <+58>:  ret
```

End of assembler dump.



ebp

Dump of assembler code for function main:

```
0x0804841d <+0>:    push  ebp
0x0804841e <+1>:    mov   ebp,esp
0x08048420 <+3>:    sub   esp,0x14
0x08048423 <+6>:    mov   DWORD PTR [ebp-0x4],0xa
0x0804842a <+13>:   mov   DWORD PTR [ebp-0x8],0x14
0x08048431 <+20>:   mov   DWORD PTR [ebp-0xc],0x0
0x08048438 <+27>:   mov   eax,DWORD PTR [ebp-0x8]
0x0804843b <+30>:   mov   edx,DWORD PTR [ebp-0x4]
0x0804843e <+33>:   add   eax,edx
0x08048440 <+35>:   mov   DWORD PTR [ebp-0xc],eax
0x08048443 <+38>:   mov   eax,DWORD PTR [ebp-0xc]
0x08048446 <+41>:   mov   DWORD PTR [esp+0x4],eax
0x0804844a <+45>:   mov   DWORD PTR [esp],0x80484f0
0x08048451 <+52>:   call  0x80482f0 <printf@plt>
0x08048456 <+57>:   leave
0x08048457 <+58>:   ret
```

End of assembler dump.

ebp = esp

Dump of assembler code for function main:

```
0x0804841d <+0>:    push  ebp
0x0804841e <+1>:    mov   ebp,esp
0x08048420 <+3>:    sub   esp,0x14
0x08048423 <+6>:    mov   DWORD PTR [ebp-0x4],0xa
0x0804842a <+13>:   mov   DWORD PTR [ebp-0x8],0x14
0x08048431 <+20>:   mov   DWORD PTR [ebp-0xc],0x0
0x08048438 <+27>:   mov   eax,DWORD PTR [ebp-0x8]
0x0804843b <+30>:   mov   edx,DWORD PTR [ebp-0x4]
0x0804843e <+33>:   add   eax,edx
0x08048440 <+35>:   mov   DWORD PTR [ebp-0xc],eax
0x08048443 <+38>:   mov   eax,DWORD PTR [ebp-0xc]
0x08048446 <+41>:   mov   DWORD PTR [esp+0x4],eax
0x0804844a <+45>:   mov   DWORD PTR [esp],0x80484f0
0x08048451 <+52>:   call 0x80482f0 <printf@plt>
0x08048456 <+57>:   leave
0x08048457 <+58>:   ret
```

End of assembler dump.

20byte

ebp = esp

Dump of assembler code for function main:

0x0804841d <+0>:	push	ebp	
0x0804841e <+1>:	mov	ebp,esp	
0x08048420 <+3>:	sub	esp,0x14	
0x08048423 <+6>:	mov	DWORD PTR [ebp-0x4],0xa	→ 10
0x0804842a <+13>:	mov	DWORD PTR [ebp-0x8],0x14	→ 20
0x08048431 <+20>:	mov	DWORD PTR [ebp-0xc],0x0	→ 0
0x08048438 <+27>:	mov	eax,DWORD PTR [ebp-0x8]	
0x0804843b <+30>:	mov	edx,DWORD PTR [ebp-0x4]	
0x0804843e <+33>:	add	eax,edx	
0x08048440 <+35>:	mov	DWORD PTR [ebp-0xc],eax	
0x08048443 <+38>:	mov	eax,DWORD PTR [ebp-0xc]	
0x08048446 <+41>:	mov	DWORD PTR [esp+0x4],eax	
0x0804844a <+45>:	mov	DWORD PTR [esp],0x80484f0	
0x08048451 <+52>:	call	0x80482f0 <printf@plt>	
0x08048456 <+57>:	leave		
0x08048457 <+58>:	ret		

End of assembler dump.

Dump of assembler code for function main:

```
0x0804841d <+0>:    push  ebp
0x0804841e <+1>:    mov   ebp,esp
0x08048420 <+3>:    sub   esp,0x14
0x08048423 <+6>:    mov   DWORD PTR [ebp-0x4],0xa
0x0804842a <+13>:   mov   DWORD PTR [ebp-0x8],0x14
0x08048431 <+20>:   mov   DWORD PTR [ebp-0xc],0x0
0x08048438 <+27>:   mov   eax,DWORD PTR [ebp-0x8]
0x0804843b <+30>:   mov   edx,DWORD PTR [ebp-0x4]
0x0804843e <+33>:   add   eax,edx
0x08048440 <+35>:   mov   DWORD PTR [ebp-0xc],eax
0x08048443 <+38>:   mov   eax,DWORD PTR [ebp-0xc]
0x08048446 <+41>:   mov   DWORD PTR [esp+0x4],eax
0x0804844a <+45>:   mov   DWORD PTR [esp],0x80484f0
0x08048451 <+52>:   call  0x80482f0 <printf@plt>
0x08048456 <+57>:   leave
0x08048457 <+58>:   ret
```

End of assembler dump.

Dump of assembler code for function main:

```
0x0804841d <+0>:    push  ebp
0x0804841e <+1>:    mov   ebp,esp
0x08048420 <+3>:    sub   esp,0x14
0x08048423 <+6>:    mov   DWORD PTR [ebp-0x4],0xa
0x0804842a <+13>:   mov   DWORD PTR [ebp-0x8],0x14
0x08048431 <+20>:   mov   DWORD PTR [ebp-0xc],0x0
0x08048438 <+27>:   mov   eax,DWORD PTR [ebp-0x8]
0x0804843b <+30>:   mov   edx,DWORD PTR [ebp-0x4]
0x0804843e <+33>:   add   eax,edx
0x08048440 <+35>:   mov   DWORD PTR [ebp-0xc],eax
0x08048443 <+38>:   mov   eax,DWORD PTR [ebp-0xc]
0x08048446 <+41>:   mov   DWORD PTR [esp+0x4],eax
0x0804844a <+45>:   mov   DWORD PTR [esp],0x80484f0
0x08048451 <+52>:   call  0x80482f0 <printf@plt>
0x08048456 <+57>:   leave
0x08048457 <+58>:   ret
```

End of assembler dump.

Dump of assembler code for function main:

```
0x0804841d <+0>:    push  ebp
0x0804841e <+1>:    mov   ebp,esp
0x08048420 <+3>:    sub   esp,0x14
0x08048423 <+6>:    mov   DWORD PTR [ebp-0x4],0xa
0x0804842a <+13>:   mov   DWORD PTR [ebp-0x8],0x14
0x08048431 <+20>:   mov   DWORD PTR [ebp-0xc],0x0
0x08048438 <+27>:   mov   eax,DWORD PTR [ebp-0x8]
0x0804843b <+30>:   mov   edx,DWORD PTR [ebp-0x4]
0x0804843e <+33>:   add   eax,edx
0x08048440 <+35>:   mov   DWORD PTR [ebp-0xc],eax
0x08048443 <+38>:   mov   eax,DWORD PTR [ebp-0xc]
0x08048446 <+41>:   mov   DWORD PTR [esp+0x4],eax
0x0804844a <+45>:   mov   DWORD PTR [esp],0x80484f0
0x08048451 <+52>:   call  0x80482f0 <printf@plt>
0x08048456 <+57>:   leave
0x08048457 <+58>:   ret
```

End of assembler dump.

Dump of assembler code for function main:

```
0x0804841d <+0>:    push    ebp
0x0804841e <+1>:    mov     ebp,esp
0x08048420 <+3>:    sub     esp,0x14
0x08048423 <+6>:    mov     DWORD PTR [ebp-0x4],0xa
0x0804842a <+13>:   mov     DWORD PTR [ebp-0x8],0x14
0x08048431 <+20>:   mov     DWORD PTR [ebp-0xc],0x0
0x08048438 <+27>:   mov     eax,DWORD PTR [ebp-0x8]
0x0804843b <+30>:   mov     edx,DWORD PTR [ebp-0x4]
0x0804843e <+33>:   add     eax,edx
0x08048440 <+35>:   mov     DWORD PTR [ebp-0xc],eax
0x08048443 <+38>:   mov     eax,DWORD PTR [ebp-0xc]
0x08048446 <+41>:   mov     DWORD PTR [esp+0x4],eax
0x0804844a <+45>:   mov     DWORD PTR [esp],0x80484f0
0x08048451 <+52>:   call    0x80482f0 <printf@plt>
0x08048456 <+57>:   leave
0x08048457 <+58>:   ret
```

End of assembler dump.



감사합니다!