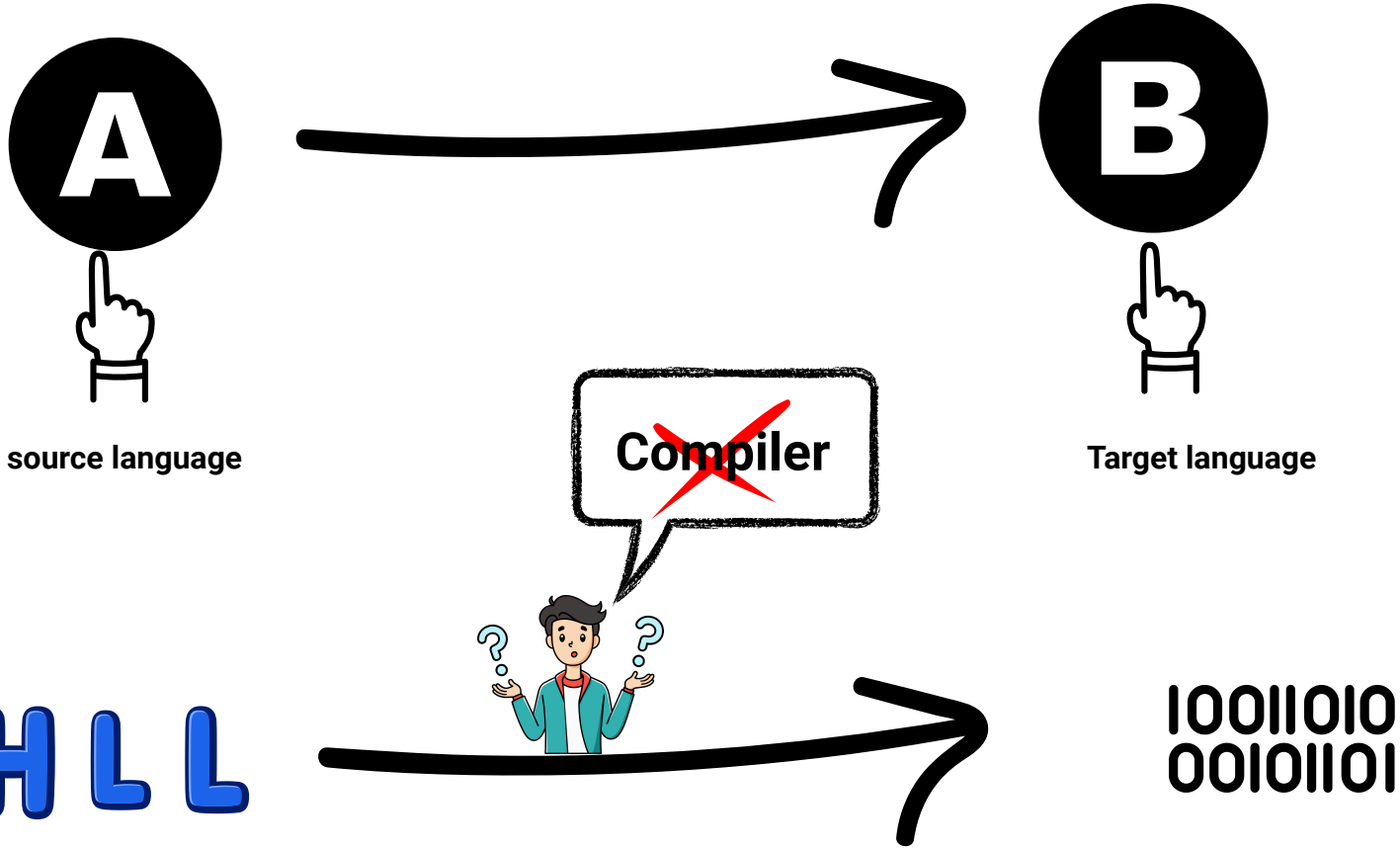
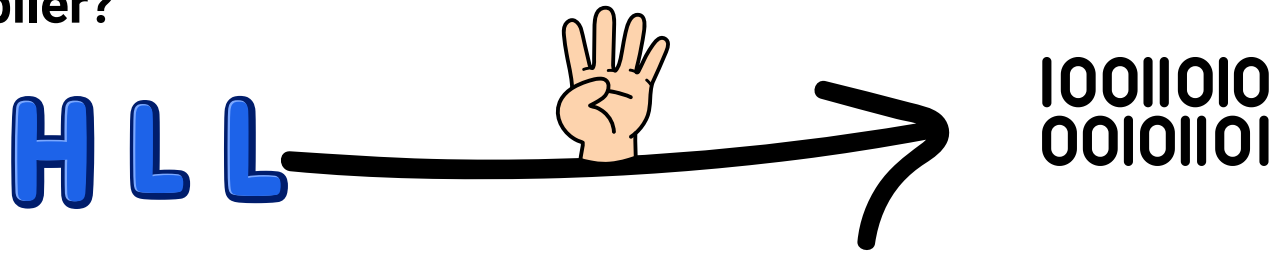
Abhinav Tiwari
osc@compilersutra.com

https://www.compilersutra.com

What is Compiler?

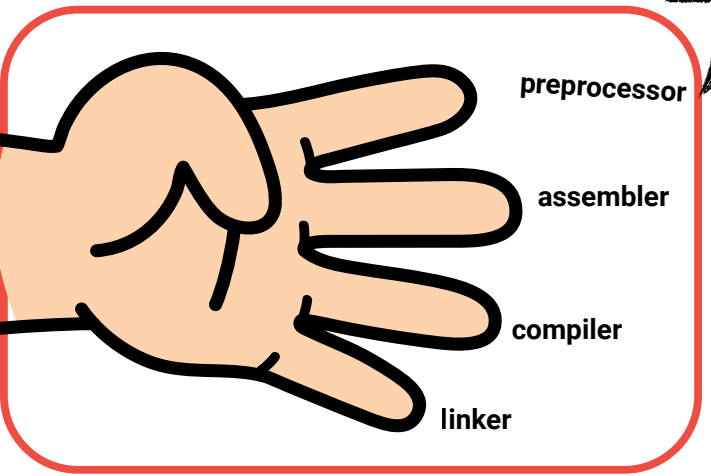


What is Compiler?



HLL —preprocessor→ HLL.i —compiler→ HLL.s —assembler→ HLL.o —linker→ HLL.elf

TOOLCHAIN



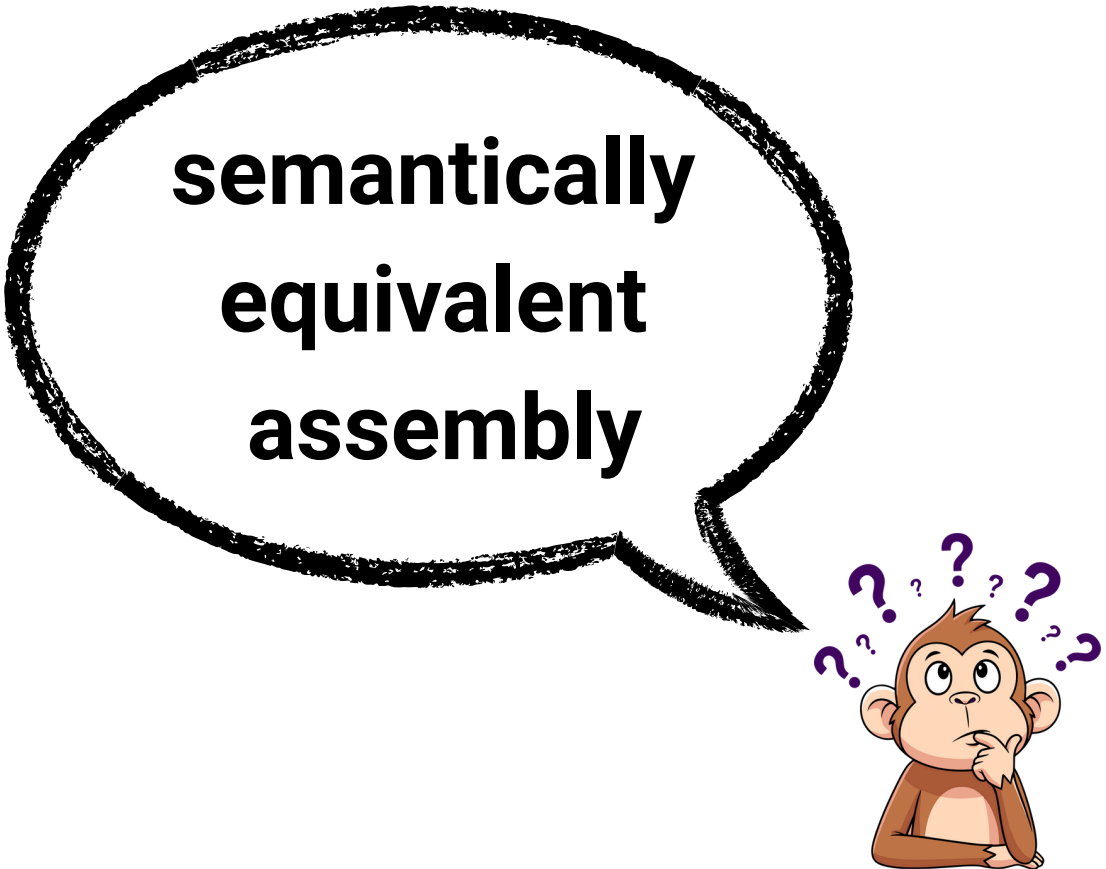
> *A **combination of multiple** tools working together in a **fixed order** to convert **source code** into a **final executable** is called a **toolchain***

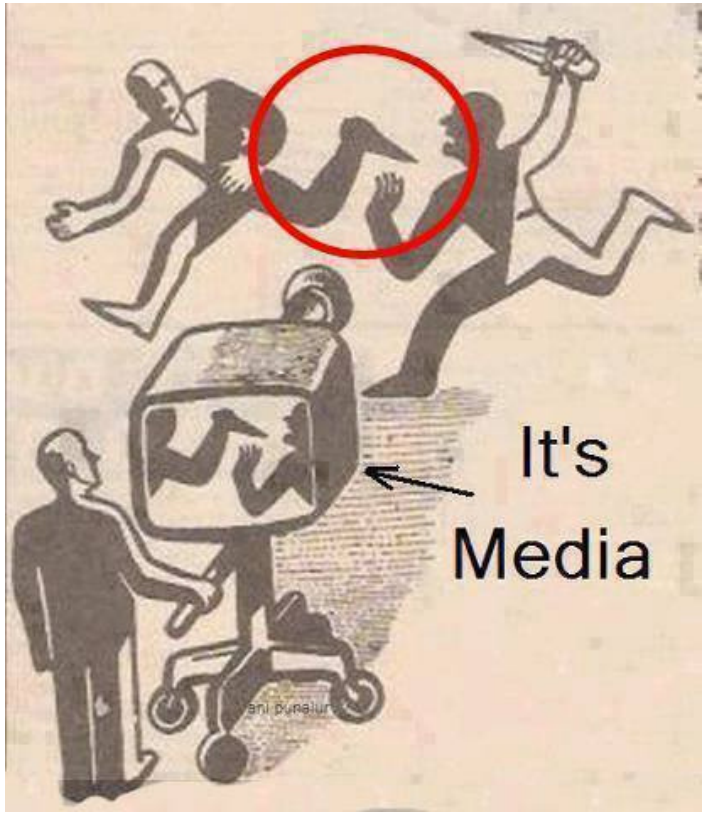

HLL
Toolchain
10011010
00101101


HLL.i
compiler
HLL.s

What is Compiler?

A compiler translates preprocessed source code into semantically equivalent assembly while performing analysis and optimizations.

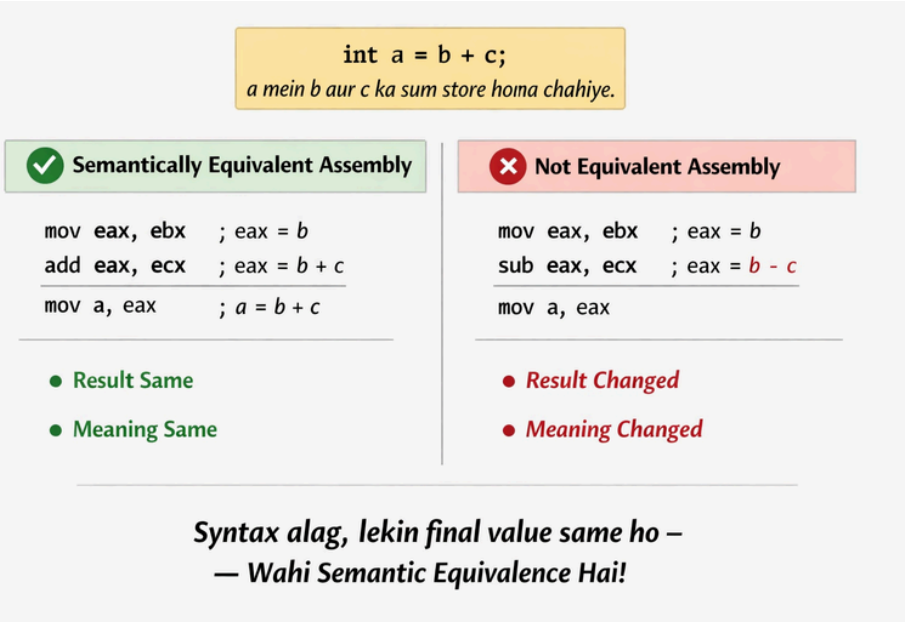



It's
Media

```
                    mov eax, ebx   ; eax = b
                    add eax, ecx   ; eax = b + c
                    mov a, eax     ; a = b + c
int a = b + c;
```

```
mov eax, ebx   ; eax = b
sub eax, ecx   ; eax = b - c
mov a, eax
```

Semantic Equivalence



Semantic equivalence means that two pieces of code produce the same result and have the same observable behavior, even if they are written differently

How to See Each Stage

How to See Each Stage

Use flag

- -save-temps

```c
#define ADD(x, y) ((x) + (y))

int foo(int b, int c) {
    int a = ADD(b, c);
    return a;
}
int main(){

        int res = foo(3,4);
        return res;
}
```

basic.c

preprocessor

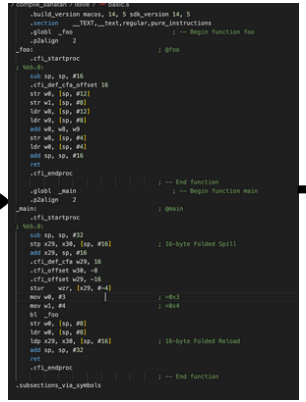
-E

```
# 1 "basic.c"
# 1 "<built-in>" 1
# 1 "<built-in>" 3
# 469 "<built-in>" 3
# 1 "<command line>" 1
# 1 "<built-in>" 2
# 1 "basic.c" 2


int foo(int b, int c) {
    int a = ((b) + (c));
    return a;
}
int main(){

 int res = foo(3,4);
 return res;
}
```

basic.i

compiler

-S



basic.s

assembler

-c

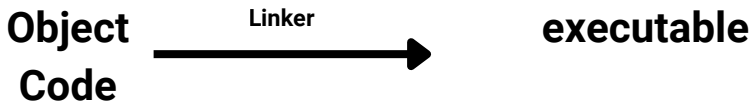
Object Code

> - *Expands macros (#define)*
> - Copies header file content (#include)
> - *Removes comments*
> - Handles #if / #ifdef / #endif

Produces expanded source (.i) for the compiler

> - Checks syntax and semantics
> - Converts preprocessed code (.i) into assembly (.s)
> - Performs optimizations
> - Generates intermediate representation (IR) internally

> - Converts assembly code (.s) into object file (.o)
> - Translates mnemonics to machine instructions



- Combines multiple object files (.o)
- Resolves undefined symbols (functions/variables)
- Links required libraries
- Produces final executable

Preprocessor → [COMPILER] → Assembler

🔒 (closed)

Next video 👇

Preprocessor → [Frontend | IR | Backend] → Assembler

🔓 (opened)