



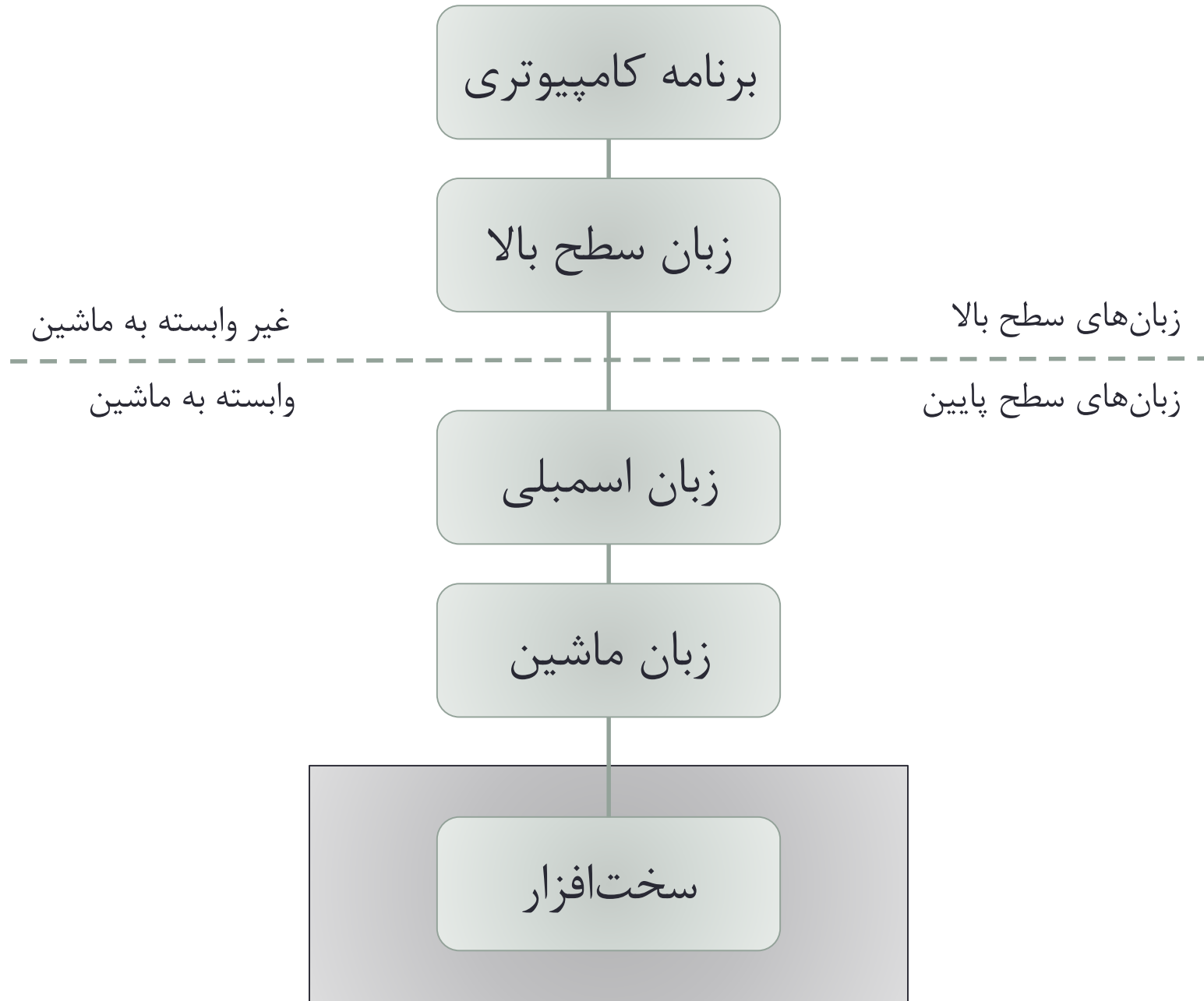
ریزپردازنده

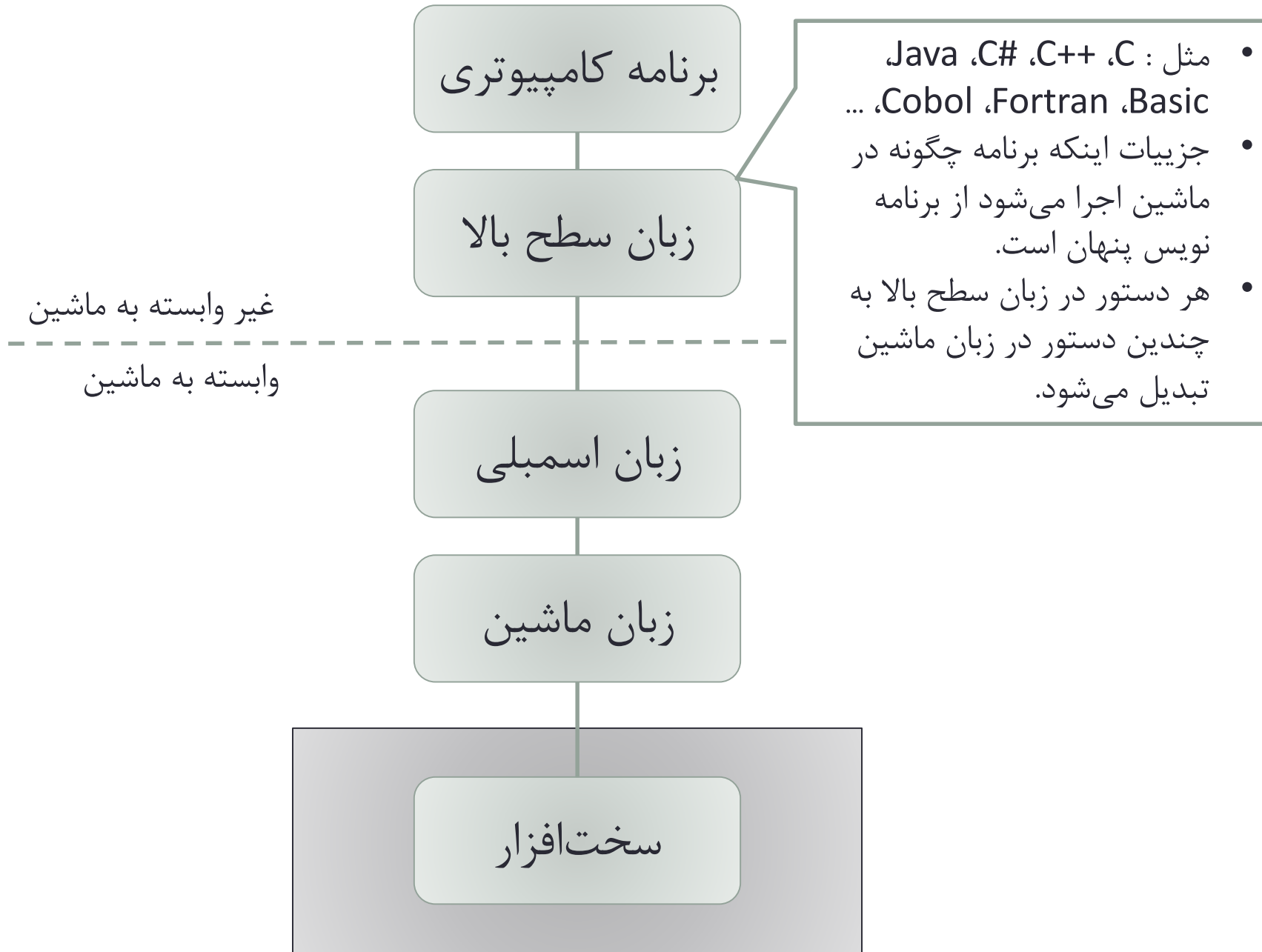
دانشکده کامپیوتر دانشگاه یزد

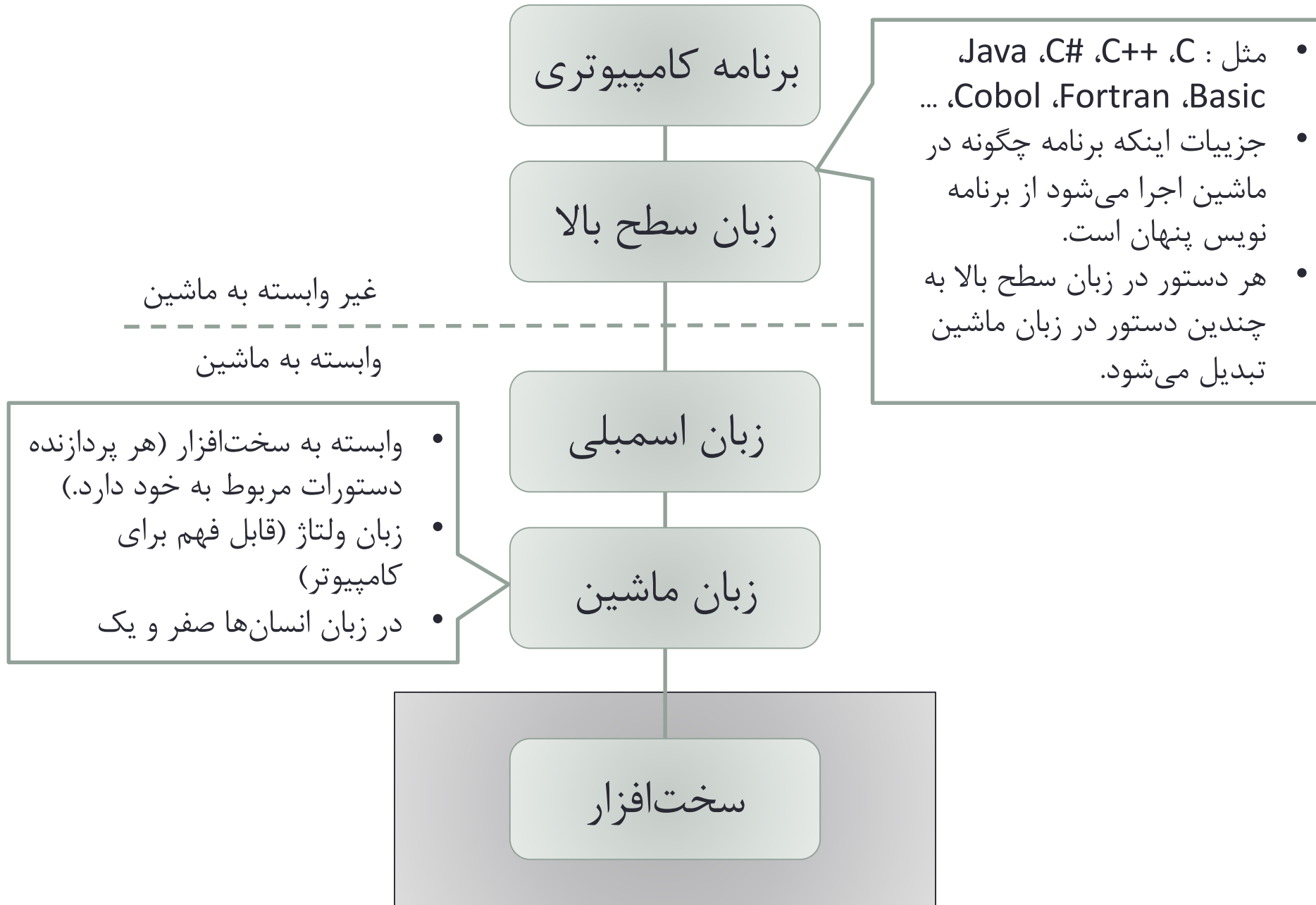
نیم سال دوم تحصیلی ۹۶-۹۷

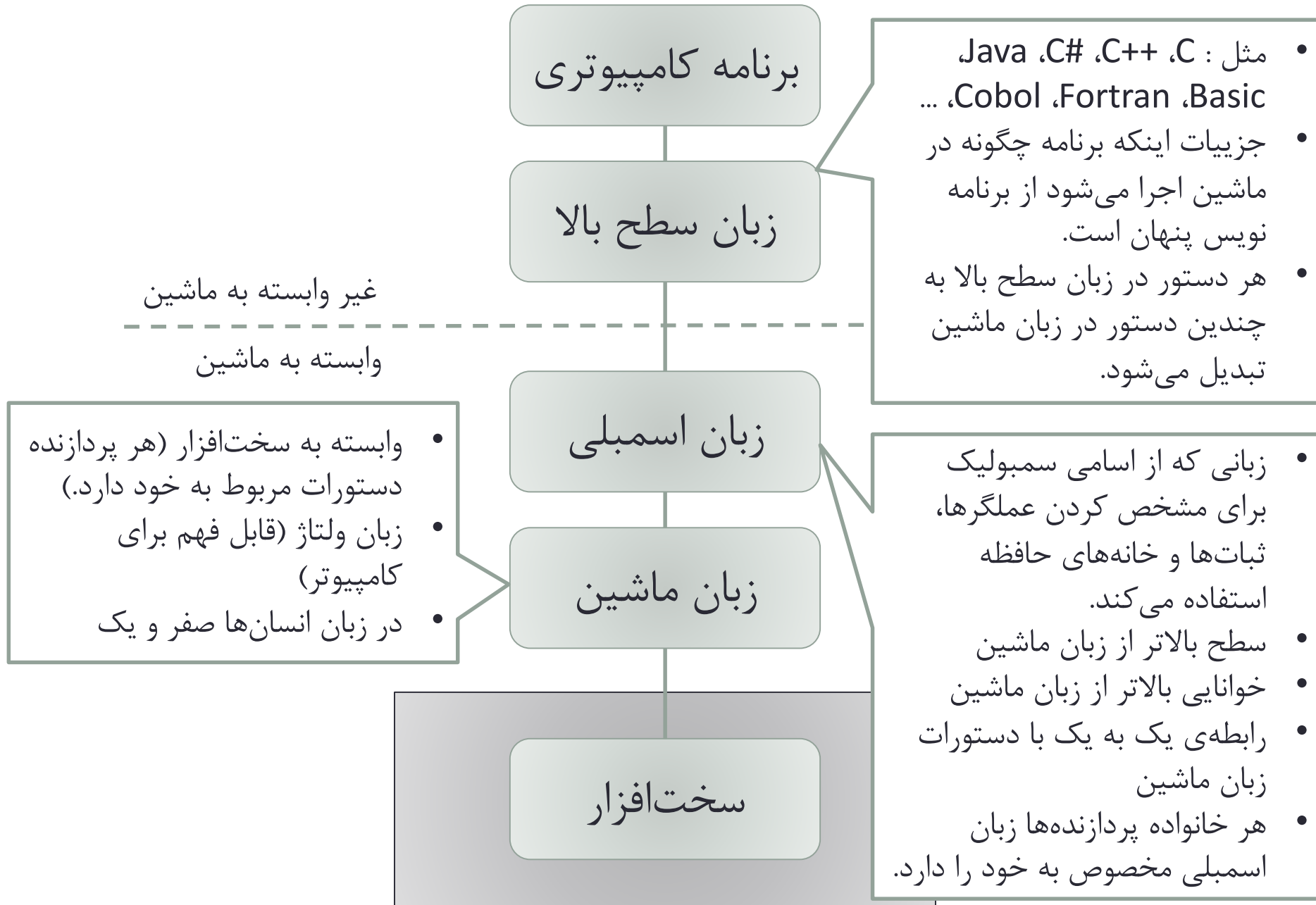
ارائه دهنده : پریسا استواری

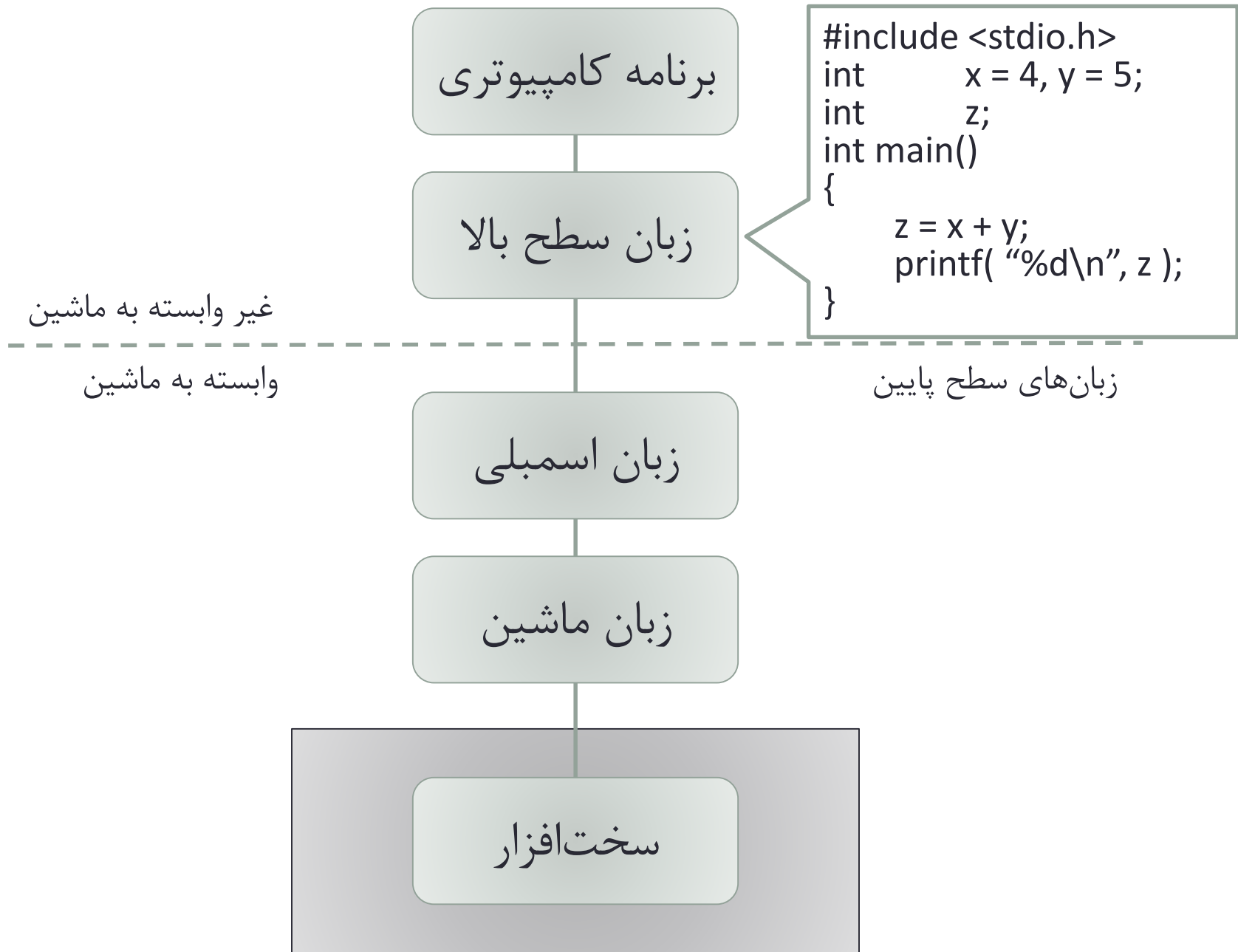
آشنایی با زبان اسمبلی

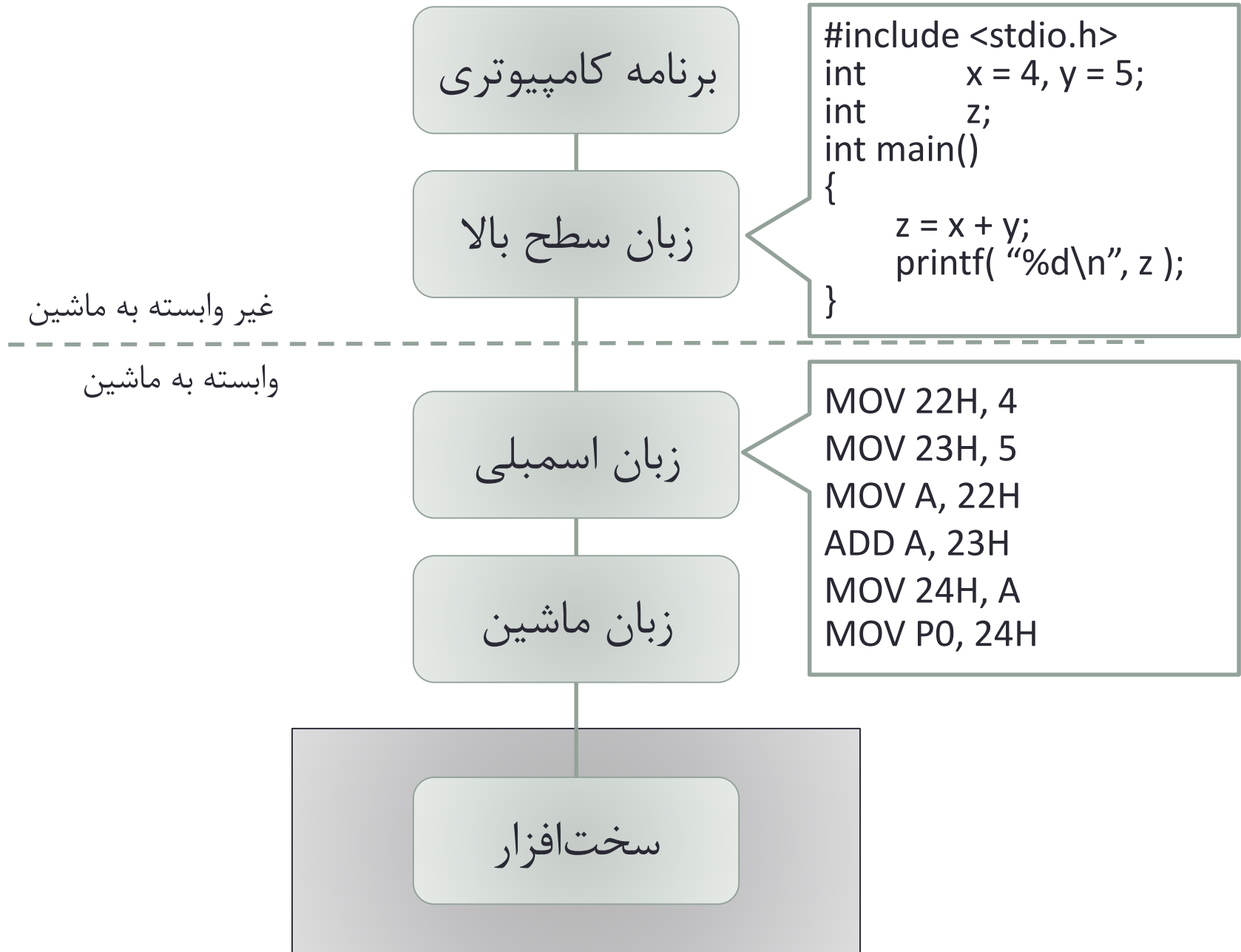


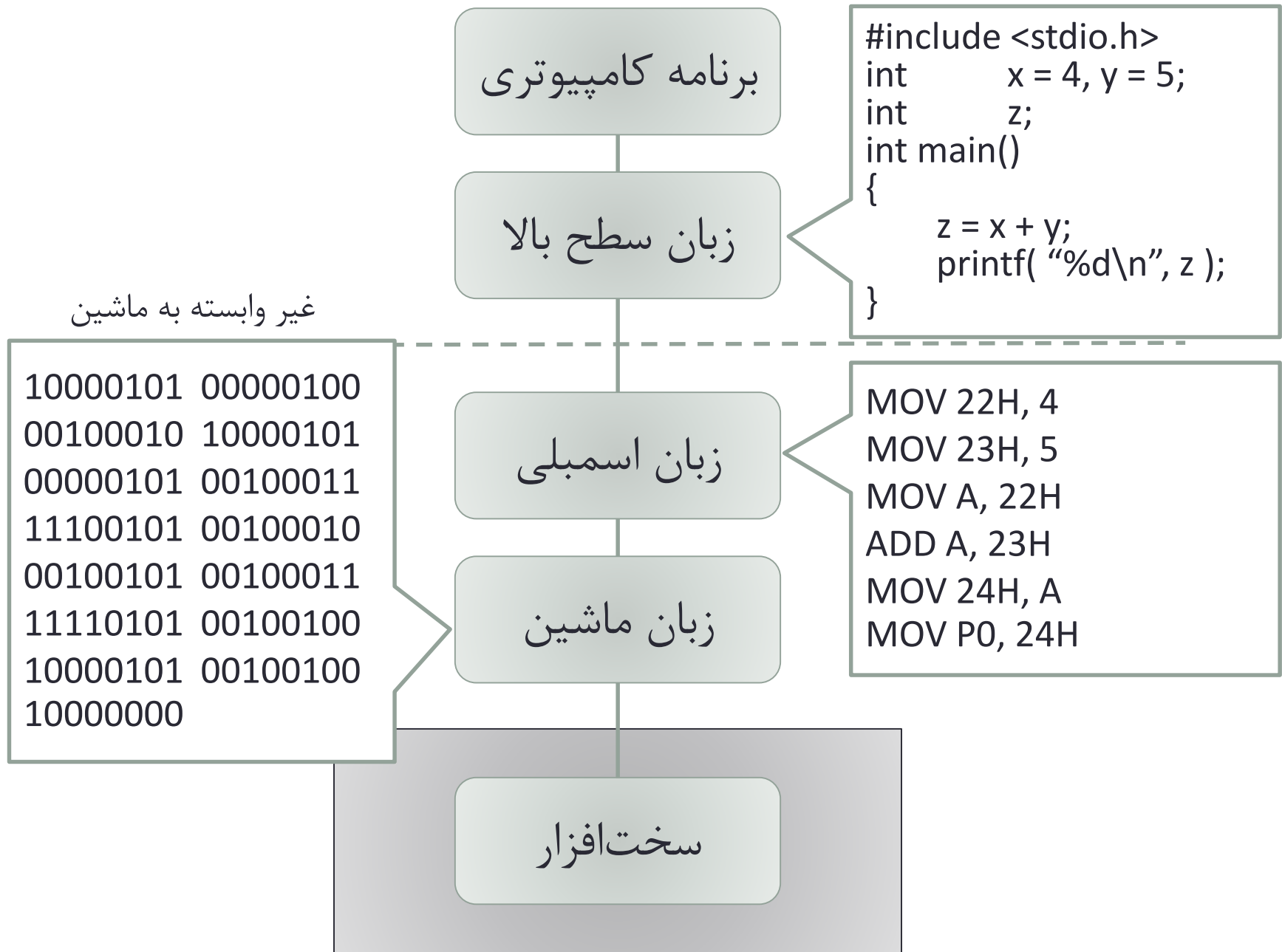












فواید استفاده از زبان‌های سطح بالا

زبان سطح پایین

```

MOV 22H, #4
MOV 23H, #5
MOV A, 22H
CJNE A, 23H, NE
EQ:  SJMP HI
NE:  JC LE
HI:  CLR C
      SUBB A, 23H
      SJMP EXIT
LE:  ADD A, 23H
EXIT: MOV 24H, A

```

زبان سطح بالا

```

int a = 4;
int b = 5;
int c;
if (a >= b)
    c = a - b;
else
    c = a + b;

```

- برنامه‌نویسی سریع‌تر است.
- به تعداد کمتری دستور نیاز است.

- عیب‌یابی برنامه ساده‌تر است.

- برنامه‌ها portable است.

- کمتر به ماشین وابسته است. بدون تغییر و یا با تغییرات کم قابل انتقال به ماشین‌های دیگر است.
- کامپایلر زبان سطح بالا را به زبان ماشین مربوطه ترجمه می‌کند.
- برنامه‌های زبان اسمبلی portable نیستند و فقط برای همان پردازنده نوشته شده‌اند.

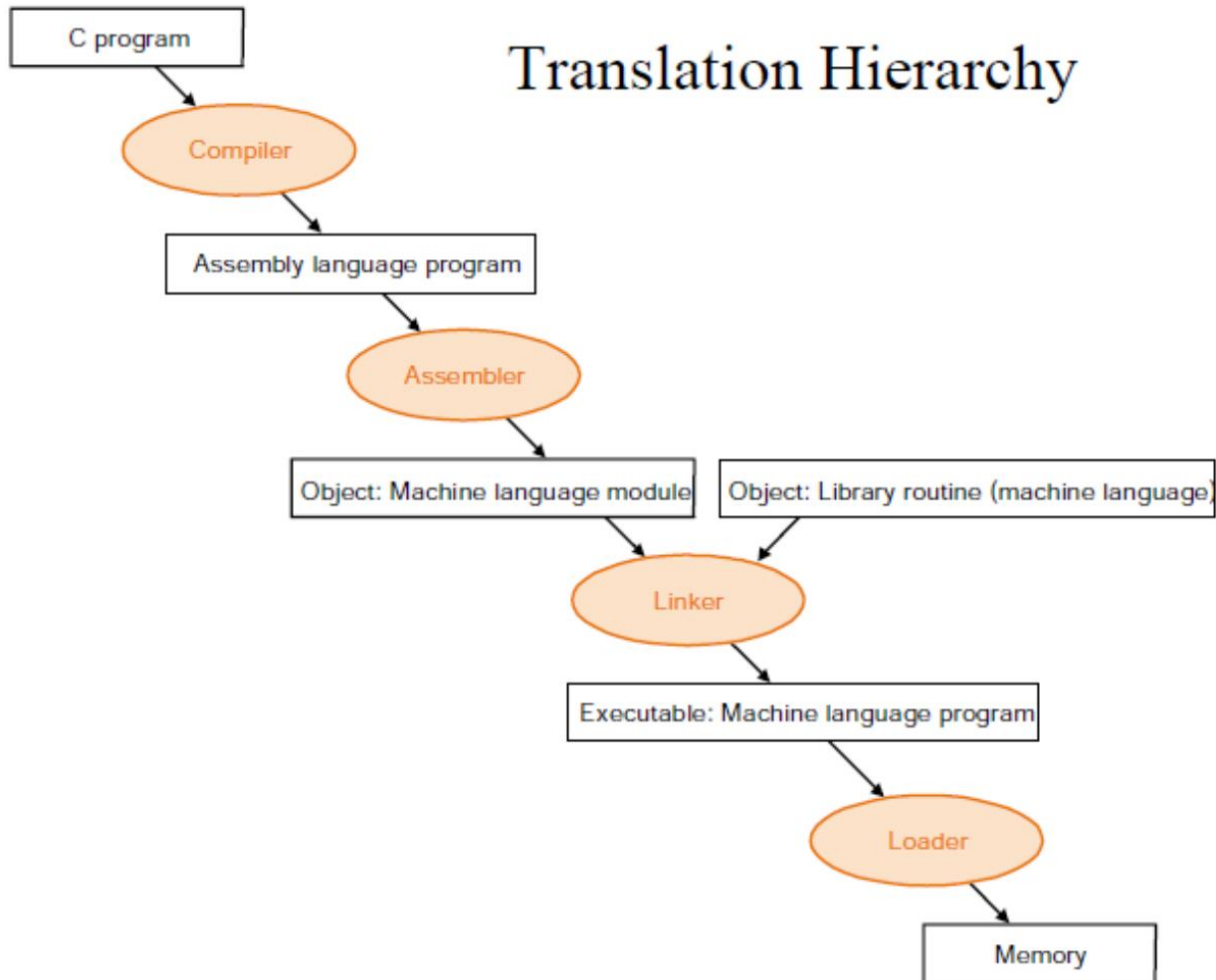
فواید زبان اسمبلی

- نشان می‌دهد برنامه چگونه با پردازنده، سیستم عامل و بایاس ارتباط برقرار می‌کند.
- نشان می‌دهد داده چگونه بر روی حافظه یا دستگاه‌های جانبی ذخیره می‌شود.
- نشان می‌دهد پردازنده چگونه دستورات را اجرا می‌کند و چگونه دستورات به داده‌ها دسترسی پیدا می‌کنند.
- نشان می‌دهد چگونه یک برنامه به دستگاه‌های خارجی دسترسی پیدا می‌کند.

دلایل استفاده از زبان اسمبلی

- یک برنامه نوشته شده به زبان اسمبلی به حافظه‌ی کمتری نیاز دارد و سرعت اجرای بالاتری نسبت به برنامه‌ی مشابه نوشته شده با زبان سطح بالا دارد.
- زبان اسمبلی به برنامه نویس توانایی نوشتن برنامه‌هایی با تکنیک بالا را می‌دهد که در زبان سطح بالا بسیار سخت و گاهی غیرممکن است.
- با وجود اینکه اغلب توسعه‌دهندگان نرم‌افزار برنامه‌های خود را به زبان سطح بالا می‌نویسند اما بعضی از قسمت‌های برنامه که از نظر زمان اجرا بحرانی هستند را به زبان اسمبلی بازنویسی می‌کنند.
- برنامه‌های سیستمی (که در زمان اجرای سایر برنامه‌ها در حافظه می‌مانند) و درایور سخت‌افزارها (برای کنترل کردن ورودی و خروجی‌ها) تقریباً همیشه به زبان اسمبلی نوشته می‌شوند.

Language Programming Tools



- Compiler
- Assembler
- Linker
- Loader
- Debugger
- Editor

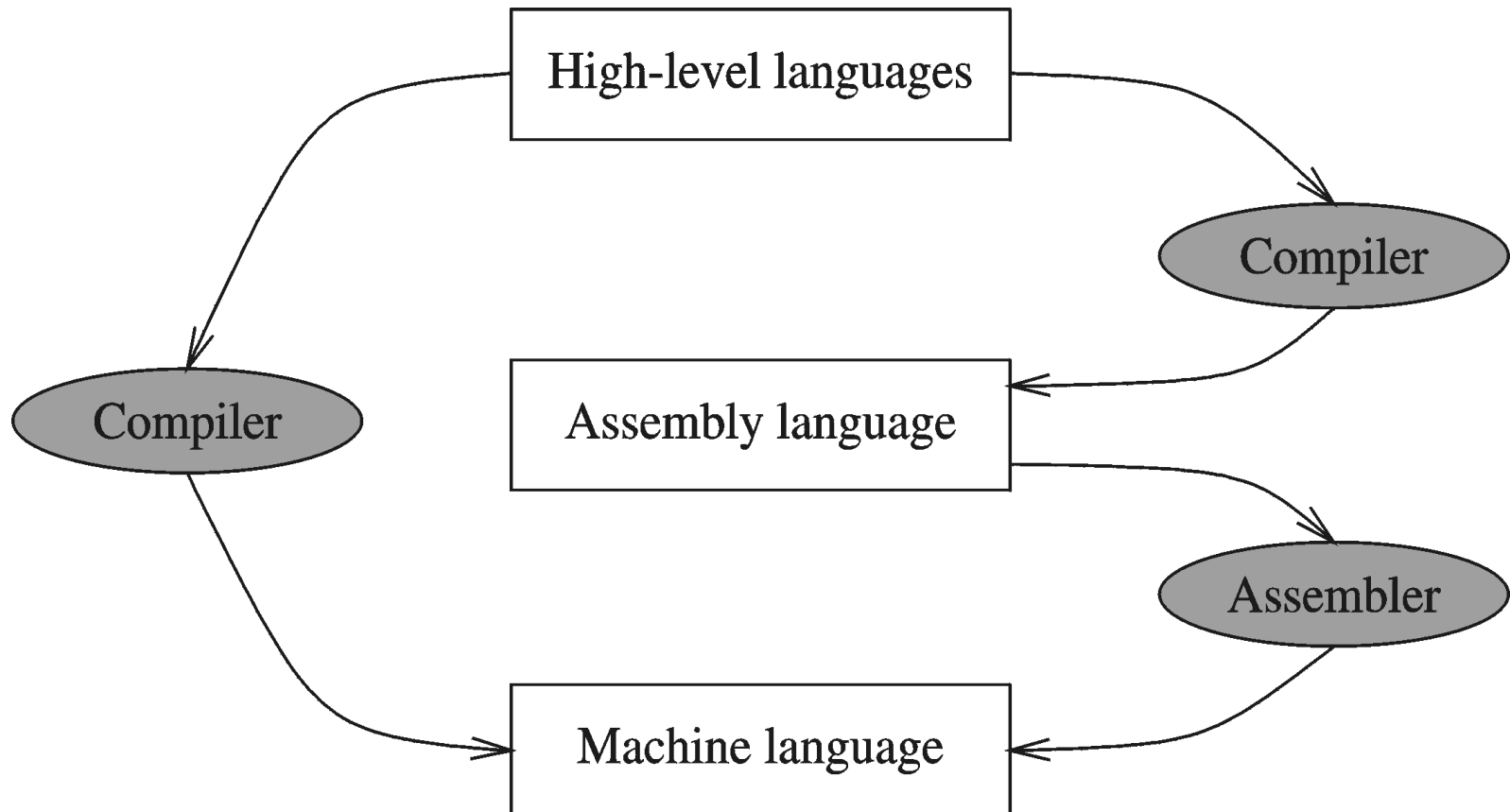
Compiler

- برنامه‌ای است که دستورات با زبان سطح بالا را به زبان اسمبلی ترجمه می‌کند.

Assembler

- اسمبلر برنامه‌ای است که کدهای نوشته شده به زبان اسمبلی را به Object file تبدیل می‌کند.
- فایل‌های Object شامل ترکیبی از دستورها، داده‌ها، و اطلاعات مورد نیاز برای قرار دادن داده‌ها در حافظه است.
- اسمبلر دستورات زبان اسمبلی را به زبان ماشین ترجمه می‌کند.
- اسمبلر اعداد دهدهی و هگزادسیمال را به اعداد باینری تبدیل می‌کند.
- اسمبلر برای ترجمه ی دستورات از دو گذار استفاده می کند :
- گذار اول : برنامه خط به خط خوانده شده و برچسب‌ها (labels) در symbol table ذخیره می‌شود.
- گذار دوم : با استفاده از اطلاعات موجود در symbol table، برای هر خط برنامه کد ماشین آن تولید می‌شود.

Assembler Vs. Compiler



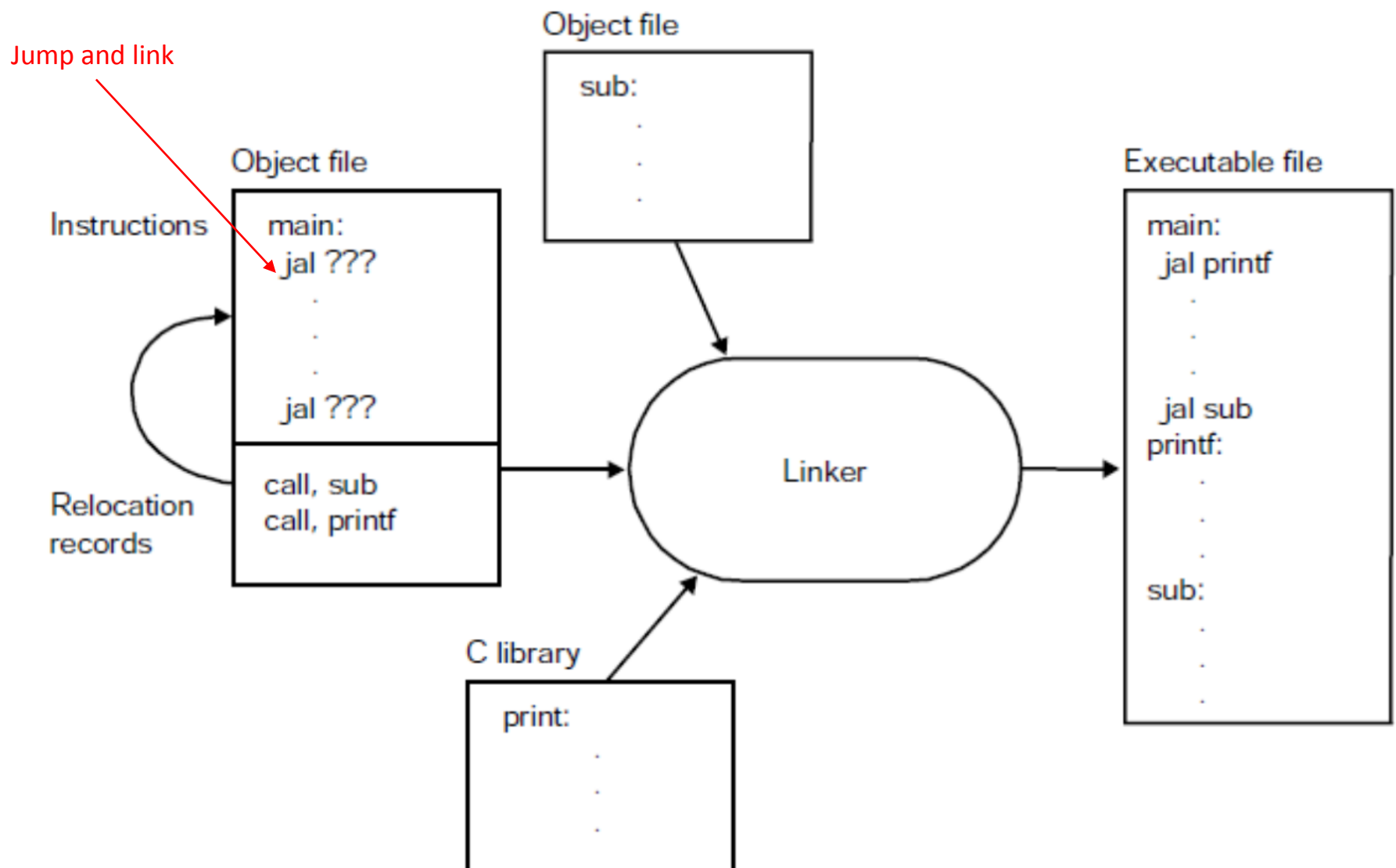
Object Files

Object file header	Text segment	Data segment	Relocation information	Symbol table	Debugging information
--------------------	--------------	--------------	------------------------	--------------	-----------------------

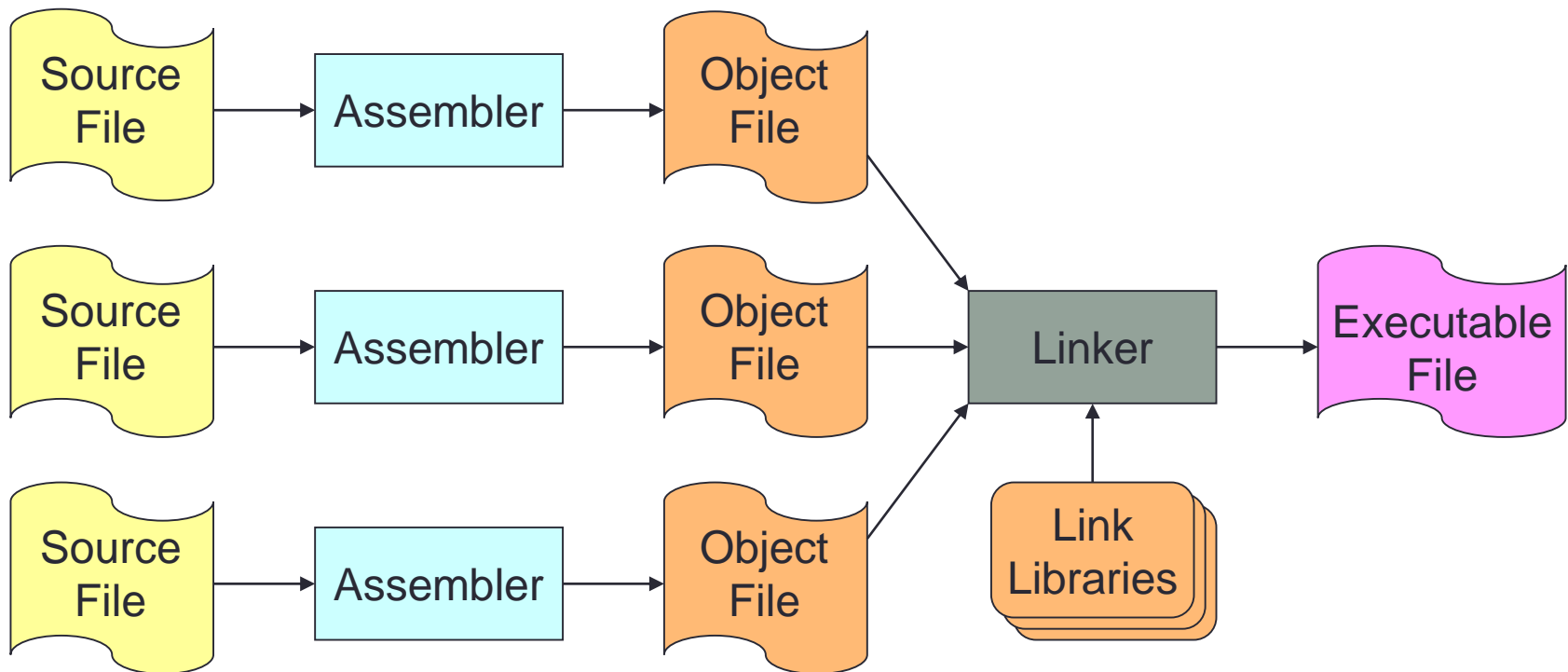
- در header سائز و مکان شروع سایر قسمت‌ها آورده می‌شود.
- در text segment دستورات به زبان ماشین قرار دارد.
- در data segment داده‌ها به صورت باینری وجود دارند.
- در relocation info دستورات و داده‌هایی که به آدرس مطلق نیاز دارند آورده می‌شوند.
- Symbol table مربوط به آدرس‌ها و برچسب‌های خارجی و لیست رفرنس‌های حل نشده است.
- در debugging info اطلاعات دیباگ وجود دارد.

Linker & Link Libraries

- برای تولید برنامه‌های قابل اجرا به برنامه‌ی Linker در کنار اسمبلر نیاز است.
- Linker فایل‌های object تولید شده توسط اسمبلر را با سایر فایل‌های object و link library ها ادغام می‌کند و یک فایل قابل اجرا می‌سازد.



Linker & Link Libraries



- یک برنامه می‌تواند از چندین برنامه تشکیل شود.
- اسمبلر هر فایل کد را به فایل object جداگانه ترجمه می‌کند.
- لینکر فایل‌های object را با link libraryها ادغام می‌کند.

Loader

- قسمتی از سیستم عامل است که یک فایل قابل اجرا را از حافظه جانبی به حافظه RAM انتقال می دهد و اجرای آن را آغاز می کند.

Debugger

- برای trace کردن برنامه
- می‌توان کد، حافظه و مقادیر رجیسترها را مشاهده کرد.

The screenshot displays the WinDbg interface with three main panes:

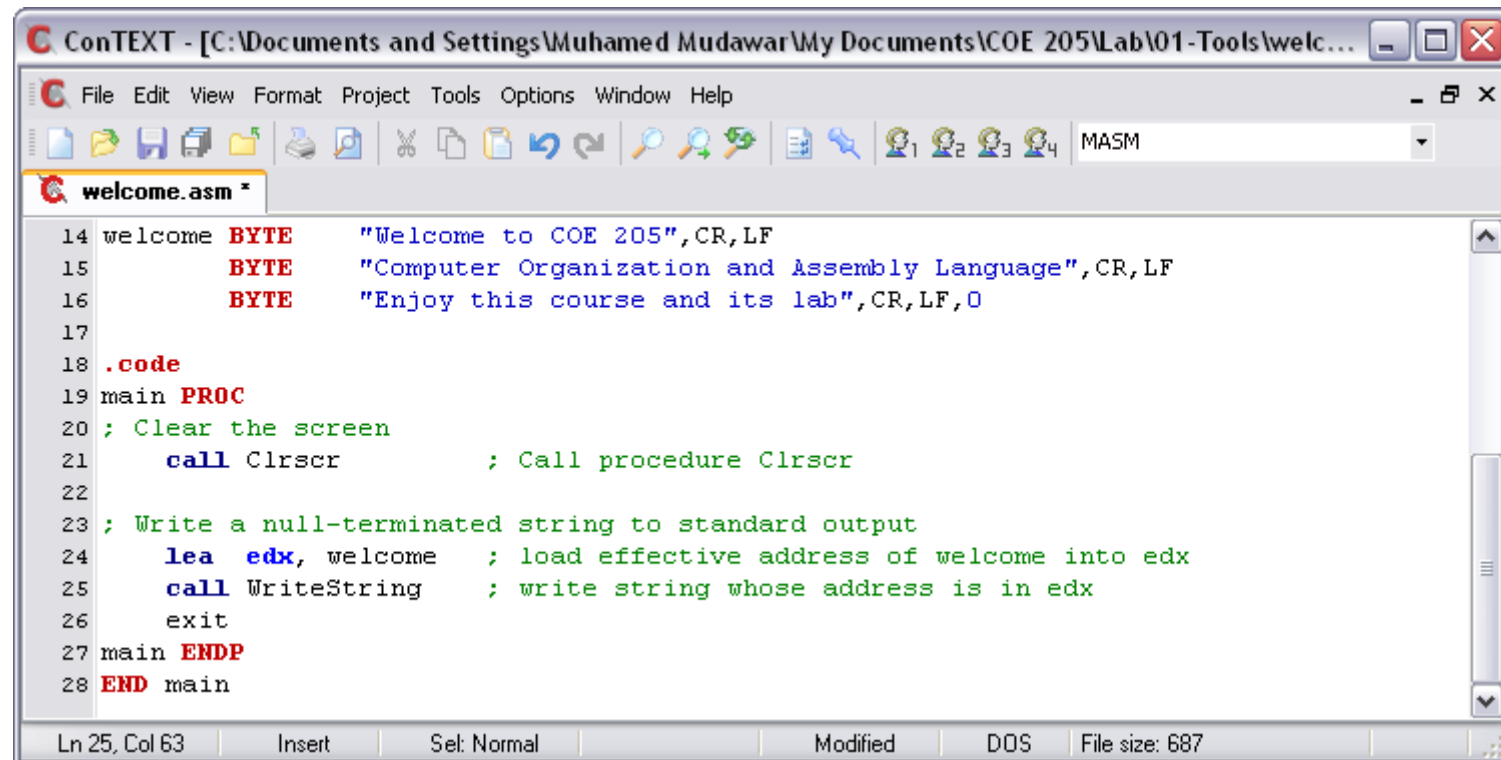
- Assembly Pane (Left):** Shows the assembly code for `Operators.exe`. The code includes data definitions and a `main` procedure demonstrating the `TYPE` operator. The instruction `mov bl, TYPE array1` is currently selected.
- Registers Pane (Middle):** A table showing the current values of the x86 registers.

Reg	Value
al	1
bl	0
cl	b0
dl	94
ax	1
bx	c000
cx	ffb0
dx	eb94
eax	1
ebx	7ffdc000
ecx	12ffb0
edx	7c90eb94
esi	fcfa9c
- Memory Pane (Right):** Shows a memory dump for `stack.exe` at virtual address `12ff94`. The display format is set to `Long Hex`. The dump shows a sequence of memory addresses and their corresponding hexadecimal values.

Virtual Address	Hex Value
0012ff94	00000000 00000fb0 00000001 00000001
0012ffa4	00000006 a1143d04 805824fd 7c90e64e
0012ffb4	7c816fd4 ffffffff 00000009 0012fff8
0012ffc4	7c816fd7 00080000 00fcfa9c 7ffd8000
0012ffd4	8054a938 0012ffc8 898463f0 ffffffff
0012ffe4	7c839aa8 7c816fe0 00000000 00000000
0012fff4	00000000 00401005 00000000 78746341

Editor

- برای ساخت source code
- تعدادی از ادیتورها دارای ویژگی‌های نمایش syntax است.



```
ConTEXT - [C:\Documents and Settings\Muhamed Mudawar\My Documents\COE 205\Lab\01-Tools\welc...
File Edit View Format Project Tools Options Window Help
welcome.asm *
14 welcome BYTE "Welcome to COE 205",CR,LF
15         BYTE "Computer Organization and Assembly Language",CR,LF
16         BYTE "Enjoy this course and its lab",CR,LF,0
17
18 .code
19 main PROC
20 ; Clear the screen
21     call Clrscr           ; Call procedure Clrscr
22
23 ; Write a null-terminated string to standard output
24     lea edx, welcome      ; load effective address of welcome into edx
25     call WriteString      ; write string whose address is in edx
26     exit
27 main ENDP
28 END main
Ln 25, Col 63   Insert   Set: Normal   Modified   DOS   File size: 687
```