برنامه سازی پیشرفته (مقدمه)

صادق اسکندری - دانشکده علوم ریاضی، گروه علوم کامپیوتر

eskandari@guilan.ac.ir

درباره این کلاس ....

کمک مدرس ها

ایمان کیانیان
(مسئول کمک مدرس ها)

امیر اصغری

عرفان محرم زاده

امیرحسین رحیمی

مینو روستا

سارا مظاهری

تینا وحیدی

علی داداش زاده

وحید عقیلی

کیان رضایی

بارم بندی نمرات

میانترم: ٤ نمره

پایانترم: ۱۰ نمره

پروژه پایانی: ۳ نمره

تمرین ها: هر کدام ۱ نمره

درباره این کلاس ....

سوالات متداول

آیا می توانم به عنوان مستمع آزاد در کلاس حضور داشته باشم؟

بله، در صورتی که فضای کلاس اجازه دهد، از حضور مستمعین آزاد استقبال می شود.

آیا تمرینات به شکل گروهی قابل انجام هستند؟

خیر، تمامی تمرینات باید به صورت انفرادی انجام شوند.

آیا پروژه های پایانی به شکل گروهی قابل انجام هستند؟

بله، پروژه های پایانی در قالب گروههای حداکثر ۳ نفری قابل انجام هستند. اطلاعات تکمیلی در صفحه پروژه های پایانی قابل دسترس هستند.

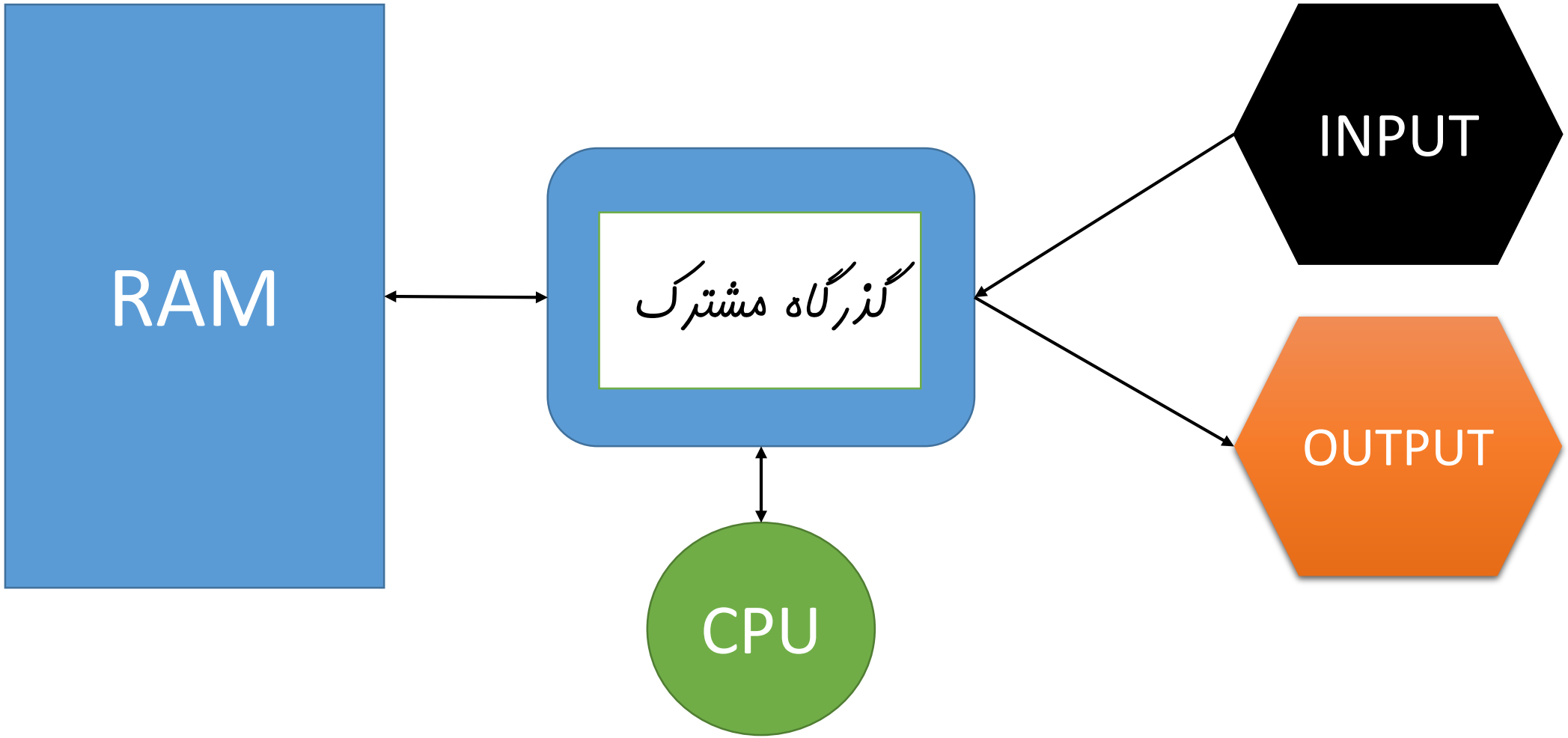
آیا می توانم برای ایمیل دانشگاهی استاد درس، پیام ارسال کنم؟

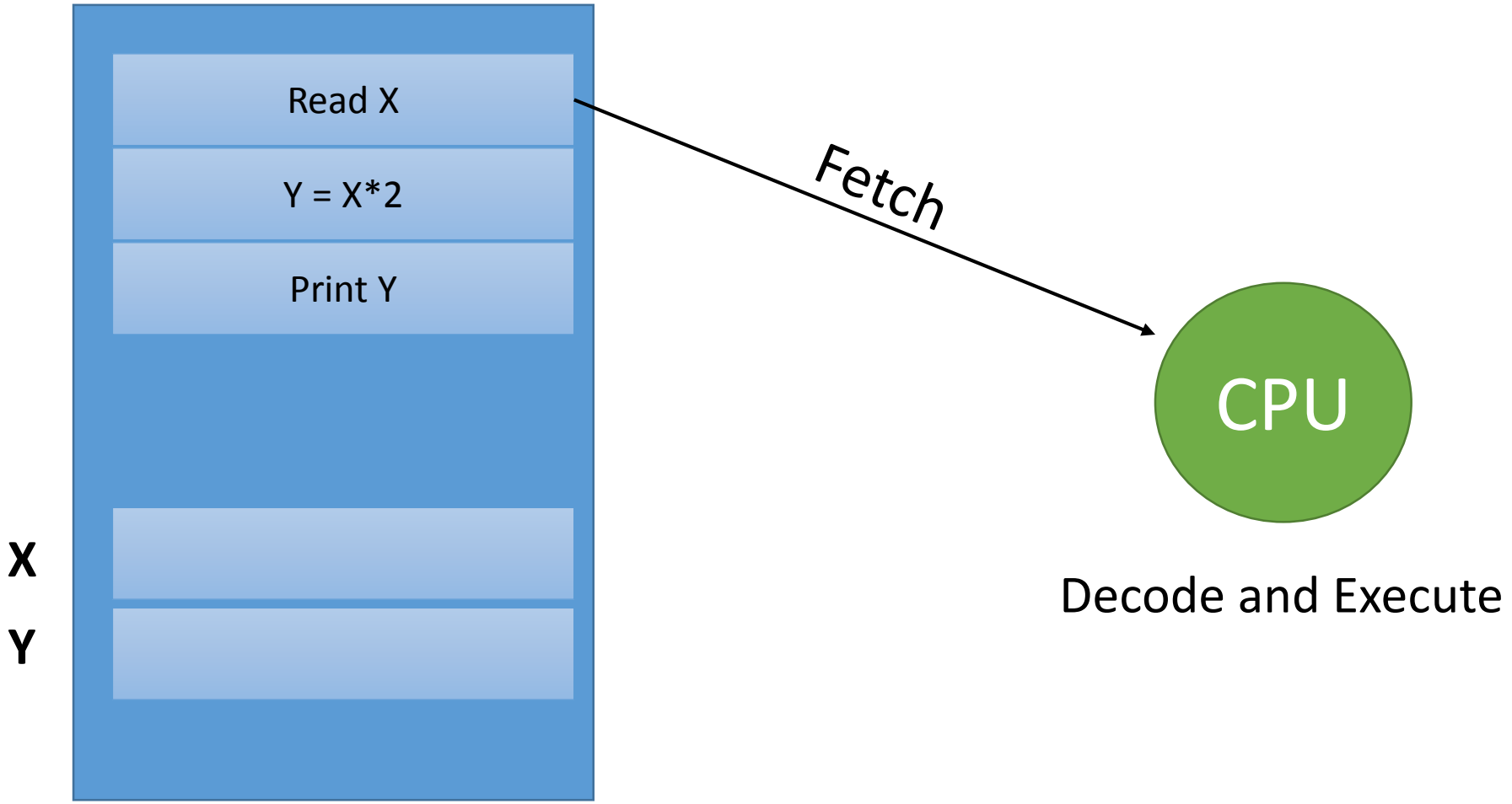
خیر، به دلیل دریافت ایمیل های فراوان، پاسخ به سوالات درسی از طریق ایمیل دانشگاهی امکان پذیر نخواهد بود.

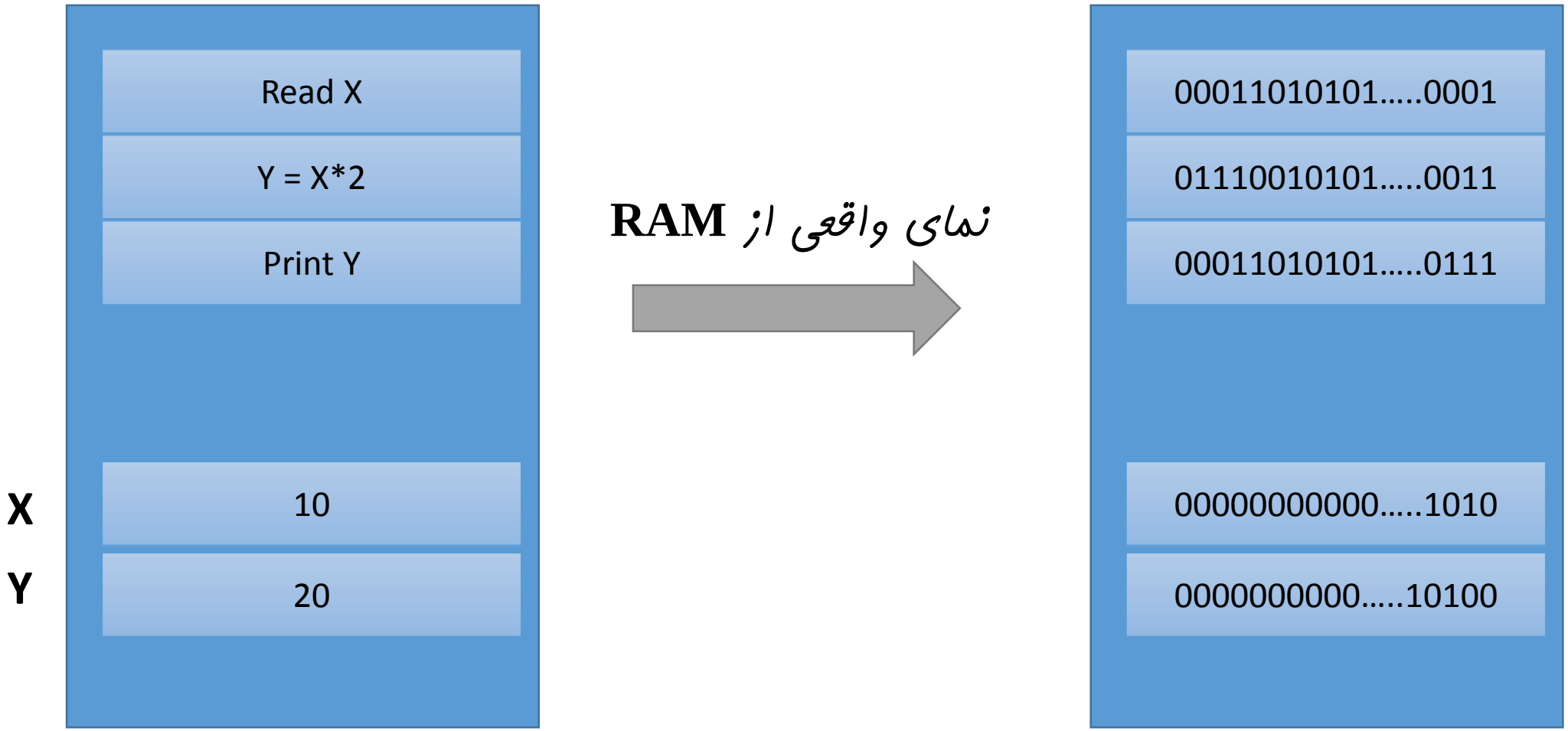
درباره این کلاس ....

https://sadegh28.github.io/AP99001/

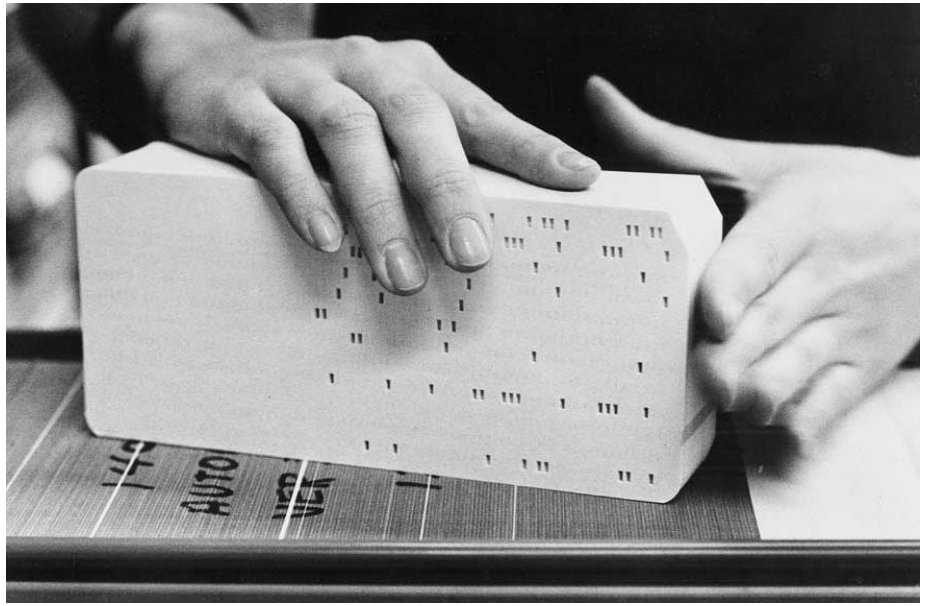
ساختار کامپیوتر پایه



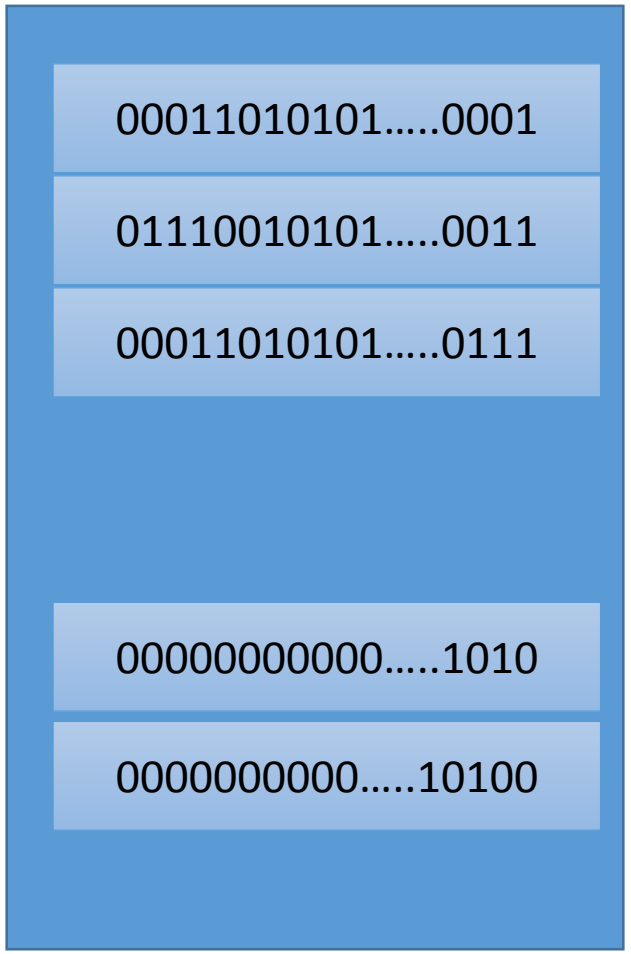

Read X
Y = X*2
Print Y
X
Y
Fetch
CPU
Decode and Execute


Read X
Y = X*2
Print Y
X
10
Y
20
نمای واقعی از RAM
00011010101…..0001
01110010101…..0011
00011010101…..0111
00000000000…..1010
0000000000…..10100

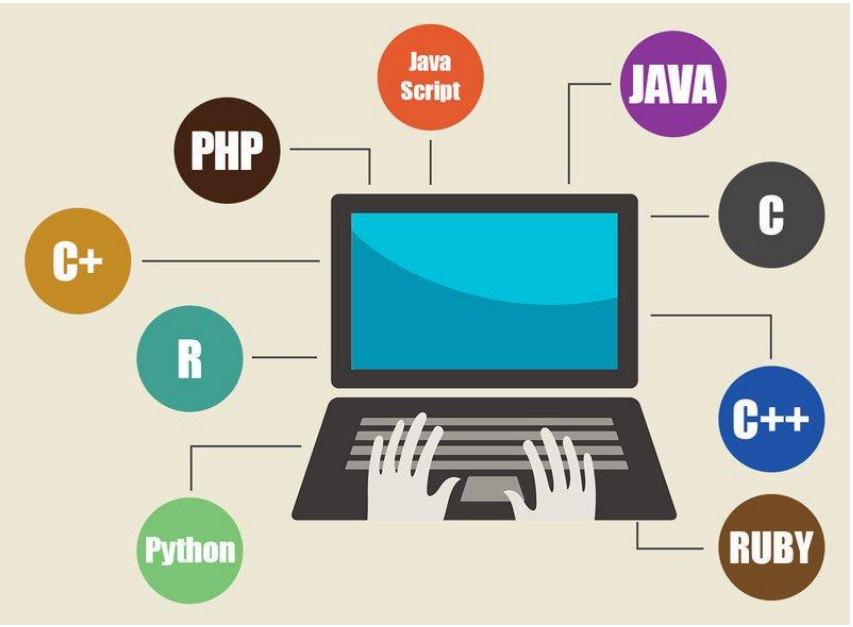
برنامه نویسی در زمان های قدیم ☹



Punch Cards



برنامه نویسی امروزه ☺



Programming Language

ترجمه

00011010101.....0001

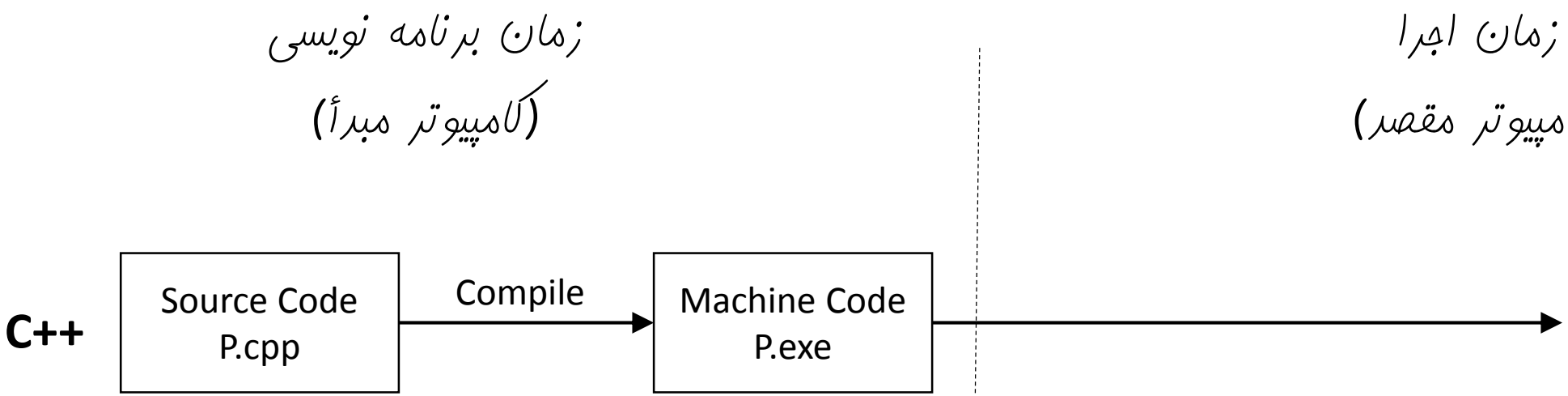
01110010101.....0011

00011010101.....0111

00000000000.....1010

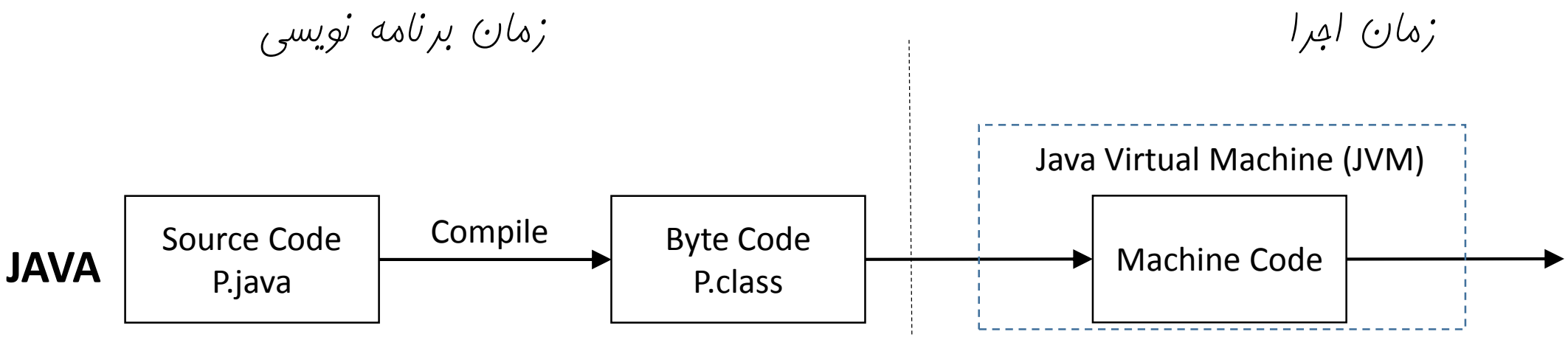
0000000000.....10100

انواع زبانهای برنامه نویسی



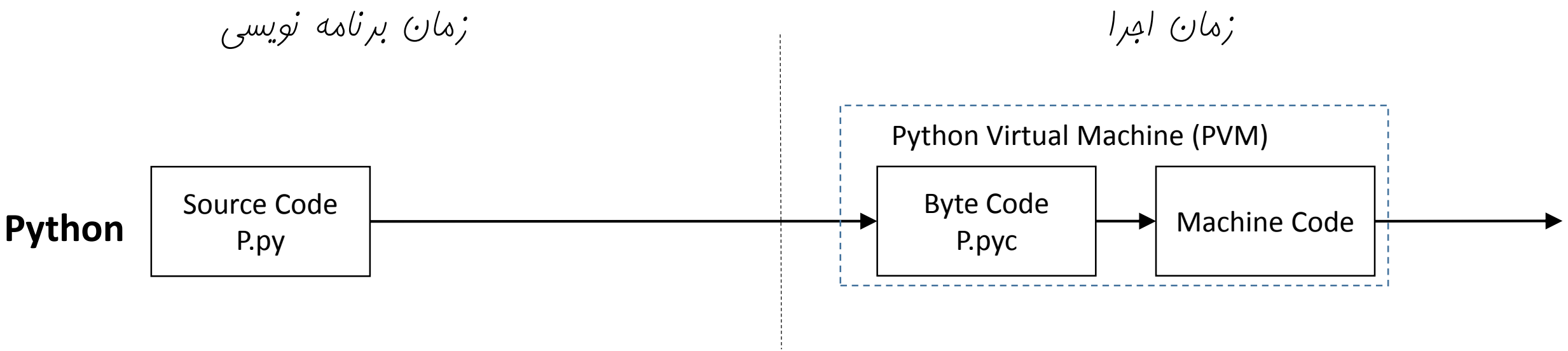
- زبان کامپایلری
- سرعت اجرای بالا ☺
- اگر معماری کامپیوتر مبدا با معماری کامپیوتر مقصد یکسان نباشد، خطا رخ می دهد ☹

انواع زبانهای برنامه نویسی



- زبان کامپایلری-مفسری
- سرعت اجرای پایین تر نسبت به زبانهای کامپایلری ☹
- اگر معماری کامپیوتر مبدا با معماری کامپیوتر مقصد یکسان نباشد، خطا رخ نمی دهد ☺

انواع زبانهای برنامه نویسی



- زبان مفسری
- سرعت اجرای پایین تر ☹
- قابلیت انتقال کدها ☺
- اگر معماری کامپیوتر مبدا با معماری کامپیوتر مقصد یکسان نباشد، خطا رخ نمی دهد ☺

- شیء گرا
- رایگان
- مفسری و قابل حمل
- نزدیک به زبان انسان
- زبان همه منظوره

آپشن ها برای برنامه نویسی در پایتون

- ## پایتون ۳ یا پایتون ۲

  اگرچه این دو نسخه تفاوت چندانی با یکدیگر ندارند،
  ما در این درس از نسخه ۳ استفاده خواهیم کرد.
  ممکن است برخی از برنامه هایی که مینویسیم در نسخه ۲
  با خطا مواجه شوند.

- ## برنامه نویسی تعاملی یا برنامه نویسی در فایل

  در مد تعاملی (Interactive Mode) نتیجه هر دستور
  در همان لحظه مشخص می شود. برای اجرای چندین
  دستورالعمل، ابتدا آنها را در یک فایل با پسوند .py
  نوشته و سپس همه را با هم اجرا می کنیم. (آشنایی با
  این دو حالت، در کلاس حل تمرین☺)

مثال

```
1  # convert.py
2  # A program to convert Celsius temps to Fahrenheit
3
4  def main():
5      celsius = eval(input ("What is the Celsius temperature? "))
6      fahrenheit = 9 / 5 * celsius + 32
7      print ("The temperature is", fahrenheit, "degrees Fahrenheit.")
8
9  main()
```

توضیحات

تعریف تابع

فراخوانی تابع

```
What is the Celsius temperature? 8
The temperature is 46.4 degrees Fahrenheit.
```

خروجی

نکات اولیه:

توضیحات (Comments): هر خطی که با # شروع شود، توسط مفسر نادیده گرفته می شود.

شناسه ها(Identifiers): هر نامی که کاربر برای بخش های مختلف برنامه خود (متغیرها، کلاسها، توابع و ... انتخاب می کند شناسه نام دارد. یک شناسه در پایتون می تواند شامل کاراکترها (بزرگ و کوچک)، اعداد و خط زیر (_) باشد به گونه ای که با عدد شروع نشود و کلمه کلیدی نباشد.

```
False      class      finally    is         return
None       continue   for        lambda     try
True       def        from       nonlocal   while
and        del        global     not        with
as         elif       if         or         yield
assert     else       import     pass
break      except     in         raise
```

کلمات کلیدی در پایتون

نکات اولیه:

پایتون نسبت به حروف بزرگ و کوچک حساس است (Case Sensitive).

استفاده از ; در انتهای دستورالعمل ها اختیاری است ولی بهتر است استفاده نشود.

```
1  # this function definition starts a new block
2  def add_numbers(a, b):
3      # this instruction is inside the block, because it's indented
4      c = a + b
5      # so is this one
6      return c
7
8  # this if statement starts a new block
9  if it_is_tuesday:
10     # this is inside the block
11     print("It's Tuesday!")
12 # this is outside the block!
13 print("Print this no matter what.")
```

بلاک ها: در جاوا و C++، بلاکهای کد به وسیله علامت های {} مشخص می شوند.

در پایتون از تورفتگی (Indentation) برای مشخص کردن بلاک ها استفاده می شود.