1. 導入

担当 寺田 実
居室 西2 510
オフィスアワー 月曜昼休み (それ以外でもメールでアポイント可)
授業ウェブページ
  http://pr.cei.uec.ac.jp/~terada/lectures/comporg/2018/

1.1 授業の進め方
(ごらんのとおり)座学
出席とります; 「出席点」はありません
(参考: 電気通信大学情報理工学部履修規程第9条
 「一つの授業科目の出席時間数が, その総授業時数の3分の2に達しない
 者には原則としてその授業科目の受験を認めない.」)

このファイル(講義メモ)をウェブページで公開
  板書に相当
  暫定版を前週金曜にアップロード(努力目標)
    ながめておくと予習になる
    ハードコピーやPC/タブレットに入れて持ってくるとよかろう
  確定版を授業後にアップロード

手書きのメモも授業後に公開

中間試験と期末試験を予定
  過去の問題をウェブページで公開してます

1.2 参考書
尾内 理紀夫「コンピュータの仕組み」(朝倉書店, 情報科学こんせぷつ 1)
丸岡 章「コンピュータアーキテクチャ」(朝倉書店, 電気・電子工学基礎シリーズ 17)
パターソン&ヘネシー「コンピュータの構成と設計」(上)(下) 日経BP社

1.3 何を学ぶか - 「計算機の仕組み」
(1)計算機の基本的構成
  構成要素
  扱うデータの形式
(2)計算機がプログラムを実行する仕組み
  機械命令とアセンブリ言語
  MIPS
(3)計算機の高速化の手法
  パイプライン処理
  キャシュメモリ
  仮想記憶

(参考)
「GPUを支える技術」(技術評論社)の序文

"本書について"
...本書では、汎用CPUの基本的な処理方法の概念は
おおよそ理解があるという想定で解説を行っています。...

"必要となる前提知識について"
...本書を読むために必要となる前提知識はそれほどありません。
以下のような基礎知識があれば、より読みやすいでしょう。
- 命令フェッチから演算に至るプロセッサの動作原理
- C言語の基礎知識
- 並列処理を行う場合の問題点

こういった予備知識を培うのが本科目の目標

1.4 準備 - プログラムに何ができるか
「基礎プログラミング」の復習
  これくらいのことはわかっていてもらいたい
  ちゃんとわかってないと, この先がたいへん

プログラム実行のモデルを頭の中に持つ

とりあえず C言語で

変数

```
----------------------
int x;
----------------------
```

変数はメモリ内の「ハコ」
  メモリって何？

配列

```
----------------------
int v[10];
----------------------
```

上例は添字は0から9まで
連続したハコがメモリ上にとられる
  連続って何が連続？

代入

```
----------------------
x = 10;
y = 20;
x = y;
y = x;
----------------------
```

ハコの中身を上書きする

式の計算

```
----------------------
x = x + 1;
----------------------
```

いろいろな演算
  加減 (+-)
  乗除 (*/), 余り(%)
  シフト (>> <<)
  比較 (== != > >= < <=)

数値のデータ型のいろいろ

```
----------------------
int m;
short n;
float a;
double b;
----------------------
```

整数型と浮動小数点数型(実数型)
「ビット数」「長さ」
「表現可能範囲」「精度」

条件分岐

```
----------------------
if(x < 0){
  x = -x;
}
----------------------
```

C言語：条件式は整数値の値
  真は「非ゼロ」、偽は「ゼロ」

繰り返し

```
----------------------
n = 0;
sum = 0;
while(n < 10){
  sum = sum + n;
  n = n + 1;
}
----------------------
```

どういう順番で実行されるか

手続きと関数
宣言のしかた

呼出のしかた
  引数
返り値
ローカル変数とは

```
----------------------
int sum(int x, int y){
  int z;     // ローカル変数
  z = x + y;
  return z;
}

int main(){
  int x;     // ローカル変数
  x = sum(10, 20);
  printf("%d\n", x);
}
----------------------
```

再帰的な関数呼び出し
recursive call

階乗を漸化式で書くと

$$factorial(n) = \begin{cases} n*factorial(n-1) & (n>0) \\ 1 & (n=0) \end{cases}$$

```
----------------------
int factorial(int n){
  if(n == 0){
    return 1;
  } else {
    return n * factorial(n-1);
  }
}

int main(){
  int m;
  m = factorial(3);
  printf("%d\n", m);
}
----------------------
```

再帰呼び出しの実行のようす

```
factorial(3)
  if(n==0)? - no
  return 3*factorial(2)
  | factorial(2)
  |   if(n==0)? - no
  |   return 2*factorial(1)
  |   | factorial(1)
  |   |   if(n==0)? - no
  |   |   return 1*factorial(0)
  |   |   | factorial(0)
  |   |   |   if(n==0)? - yes -- return 1
  |   |   return 1*1
  |   return 2*1
  return 3*2
```