

Текст программы

```
# используется для сортировки
from operator import itemgetter

class Emp:
    """Микропроцессор"""

    def __init__(self, id, name, cores, dep_id):
        self.id = id
        self.name = name
        self.cores = cores
        self.dep_id = dep_id

class Dep:
    """Компьютер"""

    def __init__(self, id, model):
        self.id = id
        self.model = model

class EmpDep:
    """
    'Микропроцессоры компьютера' для реализации
    СВЯЗИ МНОГИЕ-КО-МНОГИМ
    """

    def __init__(self, dep_id, emp_id):
        self.dep_id = dep_id
        self.emp_id = emp_id

# Компьютеры
deps = [
    Dep(1, 'MacBook Air'),
    Dep(2, 'MacBook Pro'),
    Dep(3, 'iMac 2013'),

    Dep(11, 'ASUS 2000'),
    Dep(22, 'Acer 2003'),
    Dep(33, 'Honor 2020'),
]

# Микропроцессоры
emps = [
    Emp(1, 'Pentium', 2, 11),
    Emp(2, 'Celeron', 1, 11),
    Emp(3, 'Core i3', 4, 33),
    Emp(4, 'Core i7', 2, 33),
    Emp(5, 'M1', 8, 2),
]

emps_deps = [
    EmpDep(1, 1),
    EmpDep(1, 2),
    EmpDep(1, 3),
    EmpDep(3, 4),
```

```

EmpDep(2, 5),

EmpDep(11, 1),
EmpDep(22, 2),
EmpDep(33, 3),
EmpDep(33, 4),
EmpDep(33, 5),
]

```

```

def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(e.name, e.cores, d.model)
                    for d in deps
                    for e in emps
                    if e.dep_id == d.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.model, ed.dep_id, ed.emp_id)
                           for d in deps
                           for ed in emps_deps
                           if d.id == ed.dep_id]

    many_to_many = [(e.name, e.cores, dep_name)
                     for dep_name, dep_id, emp_id in many_to_many_temp
                     for e in emps if e.id == emp_id]

    print('Задание A1')
    res_11 = sorted(one_to_many, key=itemgetter(2))
    print(res_11)

    print('\nЗадание A2')
    res_12_unsorted = []
    # Перебираем все компьютеры
    for d in deps:
        # Список микропроцессоров компьютера
        d_emps = list(filter(lambda i: i[2] == d.model, one_to_many))
        # Если в компьютере есть микропроцессоры
        if len(d_emps) > 0:
            # Кол-во ядер микропроцессоров компьютера
            d_sals = [sal for _, sal, _ in d_emps]
            # Сумма ядер микропроцессоров компьютера
            d_sals_sum = sum(d_sals)
            res_12_unsorted.append((d.model, d_sals_sum))

    # Сортировка по сумме ядер
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(res_12)

    print('\nЗадание A3')
    res_13 = {}
    # Перебираем все компьютеры
    for d in deps:
        if 'Mac' in d.model:
            # Список микропроцессоров компьютеров
            d_emps = list(filter(lambda i: i[2] == d.model, many_to_many))
            # Только название микропроцессора
            d_emps_names = [x for x, _, _ in d_emps]
            # Добавляем результат в словарь
            # ключ - компьютер, значение - список названий микропроцессоров

```

```
res_13[d.model] = d_emps_names

print(res_13)

if __name__ == '__main__':
    main()
```

Результаты выполнения

Задание A1

[('Pentium', 2, 'ASUS 2000'), ('Celeron', 1, 'ASUS 2000'), ('Core i3', 4, 'Honor 2020'), ('Core i7', 2, 'Honor 2020'), ('M1', 8, 'MacBook Pro')]

Задание A2

[('MacBook Pro', 8), ('Honor 2020', 6), ('ASUS 2000', 3)]

Задание A3

{'MacBook Air': ['Pentium', 'Celeron', 'Core i3'], 'MacBook Pro': ['M1'], 'iMac 2013': ['Core i7']}