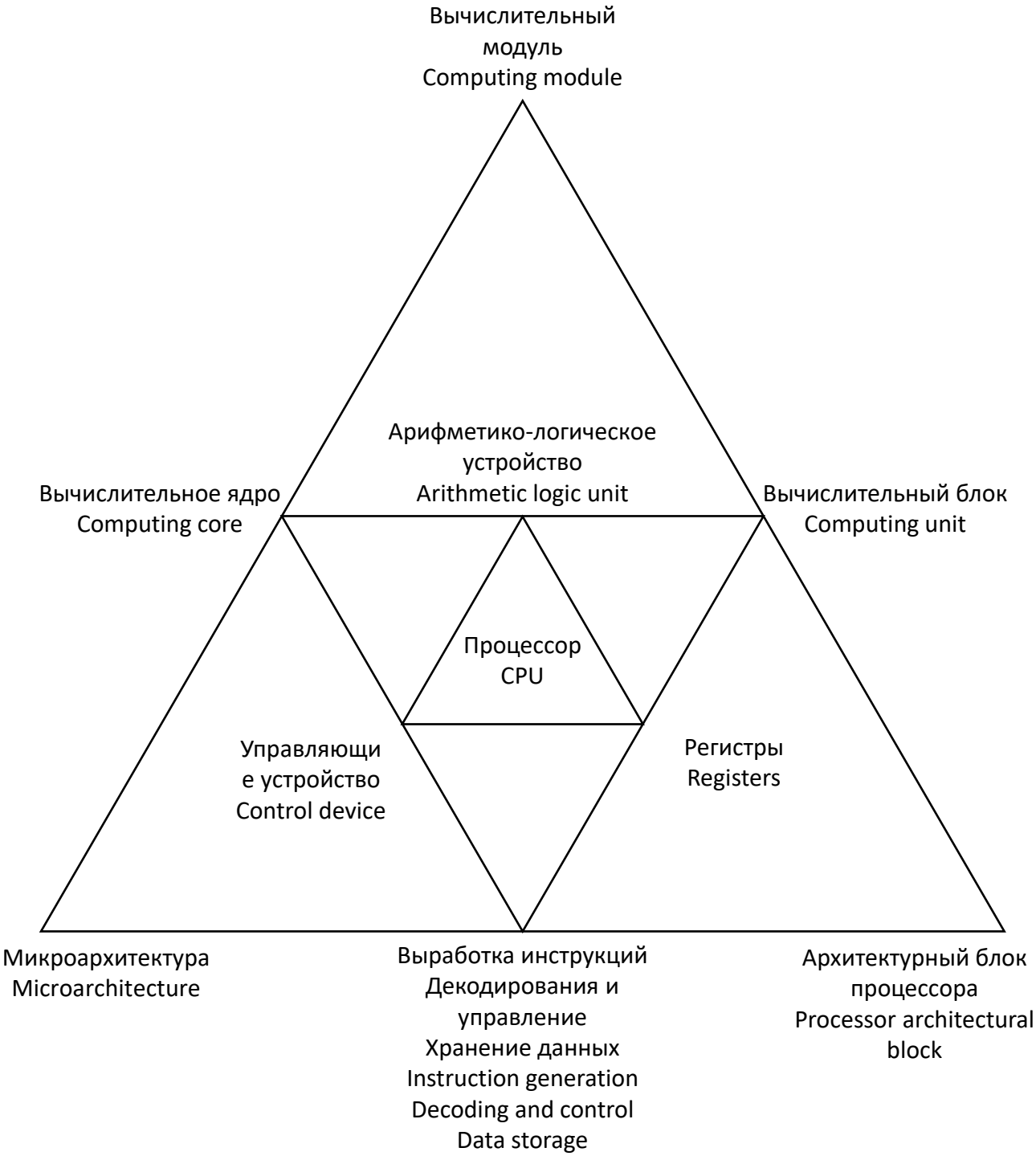
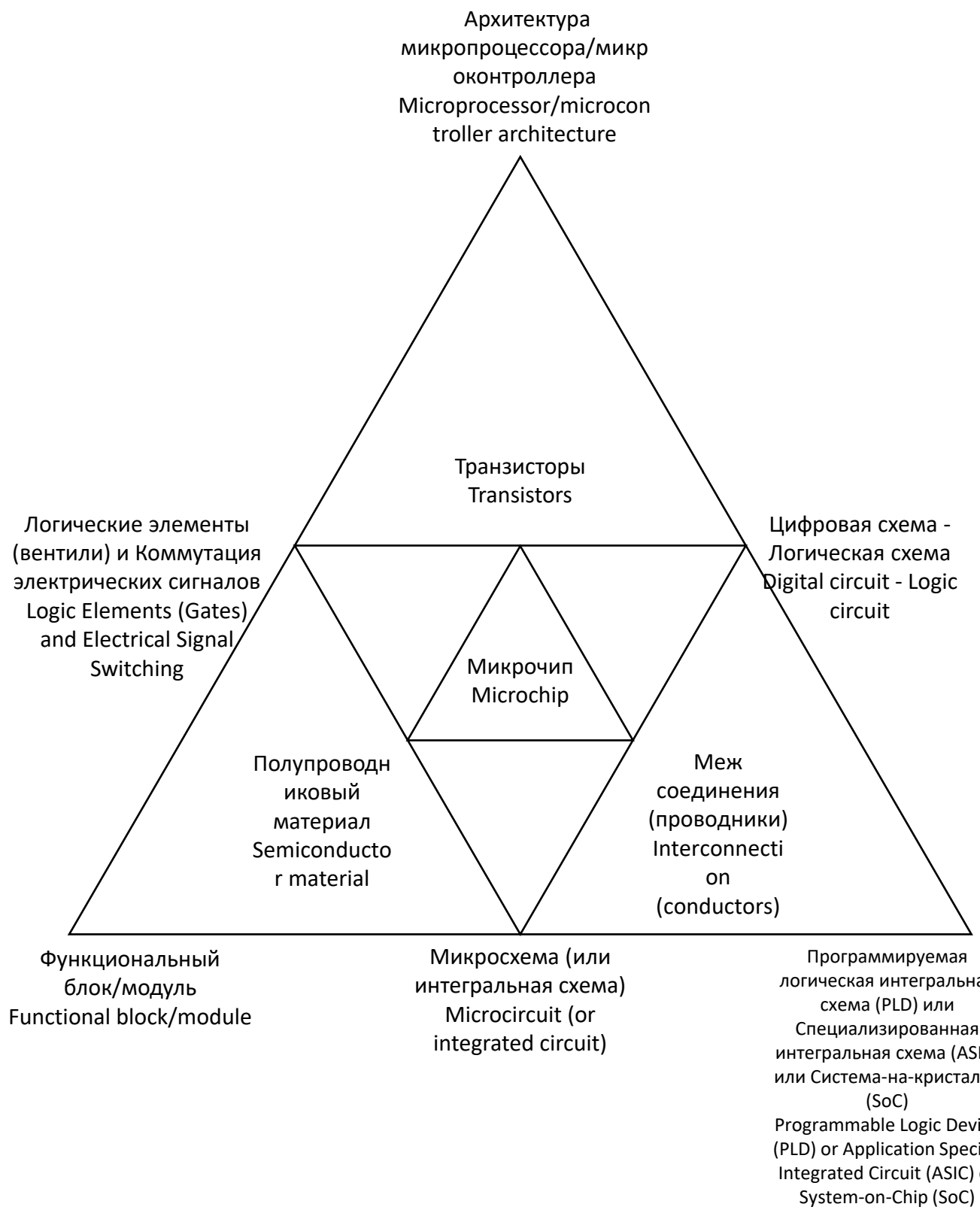
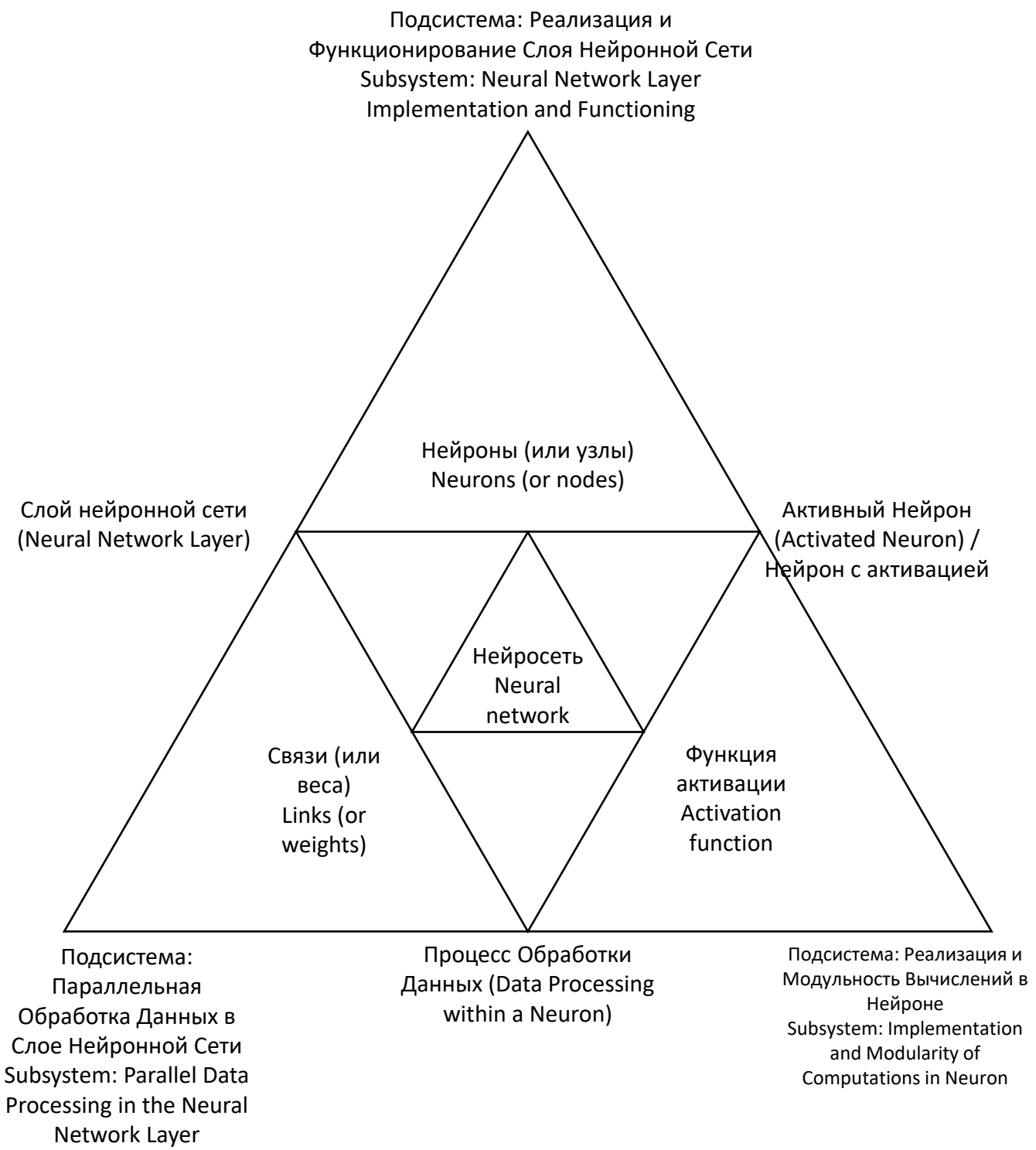


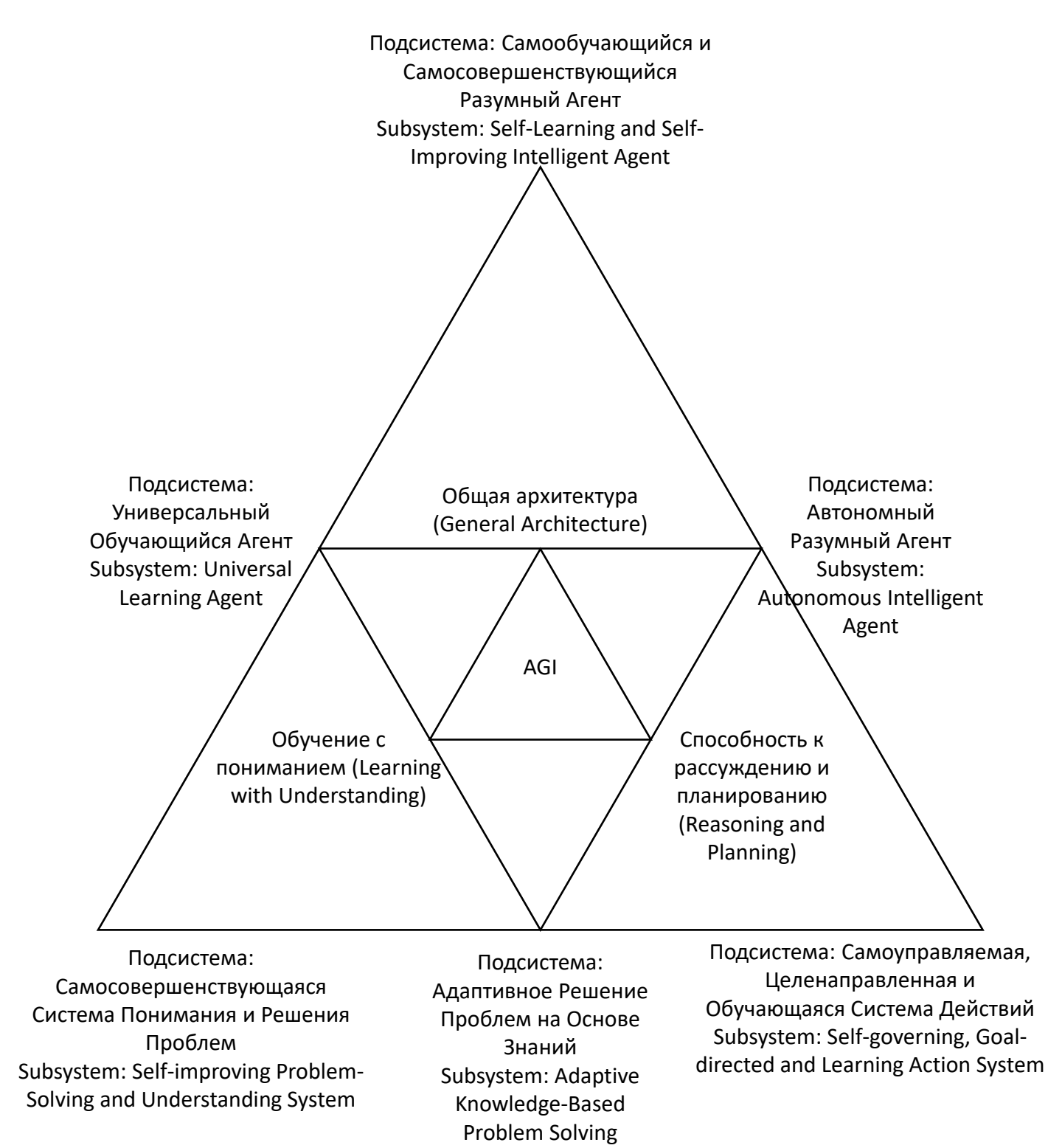
Fedorchenko Mikhail Valerevich



Fedorchenko Mikhail Valerevich



Fedorchenko Mikhail Valerevich



Fedorchenko Mikhail Valerevich

This code is a simplified version and can be extended depending on specific requirements.

```
import numpy as np

class Neuron:
    def __init__(self, weights, activation_function):
        self.weights = weights
        self.activation_function = activation_function

    def activate(self, inputs):
        total = np.dot(self.weights, inputs)
        return self.activation_function(total)

class NeuralNetworkLayer:
    def __init__(self, neurons):
        self.neurons = neurons

    def process(self, inputs):
        outputs = [neuron.activate(inputs) for neuron in self.neurons]
        return outputs

class SelfLearningAgent:
    def __init__(self, neural_network):
        self.neural_network = neural_network

    def learn(self, data):
        # Простейший пример обучения: обновление весов на основе данных
        for layer in self.neural_network:
            for neuron in layer.neurons:
                neuron.weights += np.random.rand(len(neuron.weights)) * 0.1

    def act(self, inputs):
        outputs = inputs
        for layer in self.neural_network:
            outputs = layer.process(outputs)
        return outputs

class AutonomousIntelligentAgent:
    def __init__(self, learning_agent):
        self.learning_agent = learning_agent

    def perform_task(self, task_data):
        # Пример выполнения задачи с использованием обученного агента
        result = self.learning_agent.act(task_data)
        return result

# Пример использования
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Создание нейронов и слоев нейронной сети
neurons_layer1 = [Neuron(np.random.rand(2), sigmoid) for _ in range(3)]
neurons_layer2 = [Neuron(np.random.rand(3), sigmoid) for _ in range(2)]
layer1 = NeuralNetworkLayer(neurons_layer1)
layer2 = NeuralNetworkLayer(neurons_layer2)
neural_network = [layer1, layer2]

# Создание самообучающегося агента
learning_agent = SelfLearningAgent(neural_network)

# Обучение агента на данных
data = np.random.rand(10, 2)
learning_agent.learn(data)

# Создание автономного интеллектуального агента
autonomous_agent = AutonomousIntelligentAgent(learning_agent)

# Выполнение задачи
task_data = np.random.rand(2)
result = autonomous_agent.perform_task(task_data)
print("Результат выполнения задачи:", result)
```