Межгосударственная организация высшего образования
«Российско-Армянский университет»

Утверждено
Директор Института
«___» _____ 2026, протокол №

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ

Наименование дисциплины: Архитектура ЭВМ и язык ассемблера

Авторы: к.ф.-м.н., доцент Асланян Айк Каренович,
преподаватель Авагян Тигран Суренович

Направление подготовки: 01.03.02 Прикладная математика и информатика
Наименование образовательной программы: 01.03.02 Прикладная математика и информатика

1. АННОТАЦИЯ

1.1. Краткое описание содержания данной дисциплины;

В рамках курса студенты будут изучать основы работы ЭВМ и язык ассемблера для архитектуры x86/IA-32.

После успешного освоения курса студенты должны:

- Знать основы организации ЭВМ, язык ассемблера x86/IA-32, основы организации памяти, причины появления уязвимостей в программах, физические основы работы ЭВМ.
- Уметь писать программы на языке ассемблера, анализировать причины ошибок при работе с памятью, анализировать способы улучшения производительности программ.
- Владеть навыками разработки приложений на уровне языка ассемблера.

1.2. Трудоемкость в академических кредитах и часах, формы итогового контроля – 5 кредитов (экзамен);

1.3. Взаимосвязь дисциплины с другими дисциплинами учебного плана специальности (направления):

Курс является базовым для ряда последующих курсов: Операционные системы, программирование в среде Linux

1.4. Результаты освоения программы дисциплины

Код компетенции	**Наименование компетенции**	**Код индикатора достижения компетенций**	**Наименование индикатора достижений компетенций**
ПК-1	способностью собирать, обрабатывать и интерпретировать данные современных научных исследований, необходимые для формирования выводов по соответствующим научным исследованиям	1.1	Знать принципы определения актуальности и практической значимости НИР на основе обобщения, анализа
		1.2	Уметь работать с научными источниками, проводить анализ и критически оценивать результаты

			научных исследований
		1.3	Владеть опытом выделять сильные и слабые стороны, определять значимость научных источников
ПК-7	способностью к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения	1.1	Знать методы и технологии разработки и применения системного и прикладного программного обеспечения
		1.2	Разрабатывать и применять алгоритмические и программные решения в области системного и прикладного программного обеспечения
		1.3	Владеть способностью разрабатывать и применять алгоритмические и программные решения в области системного и прикладного программного обеспечения
ПК-9	способностью составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы	1.1	Знать план выполняемой работы в ходе практической деятельности для получения профессиональных умений и опыта профессиональной деятельности
		1.2	Уметь составлять и контролировать план выполняемой работы,

			планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы
		1.3	Регулярная оценка и анализ результатов выполненной работы с целью выявления достижений, а также выявления возможных областей для улучшения и развития профессиональных навыков.
ОПК-2	Способен использовать и адаптировать существующие математические методы и системы программирования для разработки и реализации алгоритмов решения прикладных задач	1.1	Знает основные задачи и области применения математических методов, основные принципы математического моделирования, методы построения и анализа математических моделей
		1.2	Умеет выбирать математические методы, адаптировать и использовать их для разработки и реализации алгоритмов решения прикладных задач
		1.3	Осуществляет выбор и адаптацию математических методов и систем программирования для разработки и реализации

			алгоритмов решения прикладных задач
ОПК-4	Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности	1.1	Знает структуру базовых и специализированных информационных технологий, принципы их работы
		1.2	Умеет выбирать информационные технологии для решения задач профессиональной деятельности и обосновывать свой выбор
		1.3	Владеет навыками применения базовых и специализированных информационных технологий для решения задач профессиональной деятельности

2. УЧЕБНАЯ ПРОГРАММА

2.1. Цели и задачи дисциплины

Цель данного курса научить студентов основам организации ЭВМ, объяснить, как взаимодействуют между собой разные части ЭВМ, как программа, написанная на языке программирования, выполняется на ЭВМ.

2.2. Трудоемкость дисциплины и виды учебной работы (в академических часах и зачетных единицах)

Виды учебной работы	**Всего, в акад. часах**	**Распределение по семестрам**					
		___ сем	**II сем**	**___ сем**	**___ сем.**	**___ сем**	**____ сем.**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1.Общая трудоемкость изучения	**216**						

дисциплины по семестрам, в т. ч.:							
1.1. Аудиторные занятия, в т. ч.:	**64**		**64**				
1.1.1.Лекции	**32**		**32**				
1.1.2.Практические занятия, в т. ч.	**32**		**32**				
1.2. Самостоятельная работа, в т. ч.:	**80**		**80**				
1.3. Другие методы и формы занятий	**36**		**36**				
Итоговый контроль (Экзамен, Зачет, диф. зачет - указать)			**Экзамен**				

2.3. Содержание дисциплины

2.3.1. Тематический план и трудоемкость аудиторных занятий (модули, разделы дисциплины и виды занятий) по рабочему учебному плану

Разделы и темы дисциплины	**Всего (ак. часов)**	**Лекции (ак. часов)**	**Практ. Занятия (ак. часов)**	**Семинары (ак. часов)**	**Лабор. (ак. часов)**
1	**2=3+4+5+6+7**	**3**	**4**	**5**	**6**
Тема 1. Регистры процессора IA-32, размещение переменных в памяти.	**8**	**4**	**4**		
Тема 2.Основные арифметические команды, регистр флагов, переполнение, команды изменения естественного порядка выполнения программы	**10**	**4**	**6**		
Тема 3. Архитектура ЭВМ, структура ассемблерной программы	**4**	**4**			
Тема 4.Реализация управляющих операторов языка Си: условная передача данных, организация циклов.	**7**	**1**	**6**		
Тема 5. Размещение указателей, массивов, матриц и структур в памяти.	**7**	**3**	**4**		
Тема 6. Этапы компиляции. Размещение программы в памяти. Структура файлов ELF.	**4**	**4**			
Тема 7. Стек и куча	**4**	**2**	**2**		
Тема 8. Организация вызовов функций, стековый кадр функции. Оптимизация хвостового вызова. Вызов внешних функций.	**14**	**4**	**10**		
Тема 9. Числа с фиксированной и плавающей запятой.	**2**	**2**			
Тема 10. Аппаратура ЭВМ. Физические основы, логические вентили, полусумматор, полный сумматор, АЛУ.	**2**	**2**			
Тема 11. Статическая память. SR-защелка. D-защелка. D-триггер. Регистр.	**2**	**2**			
ИТОГО	**64**	**32**	**32**		

2.3.2. Краткое содержание разделов дисциплины в виде тематического плана

Тема 1. Регистры процессора IA-32, размещение переменных в памяти.

Регистры процессора x86/IA-32, 8-, 16- и 32-битные регистры. Размещение программы в памяти, сегменты кода, стека, кучи. Директивы, представление отрицательных чисел.

В.И. Юров [3]

Тема 2. Основные арифметические команды, регистр флагов, переполнение, команды изменения естественного порядка выполнения программы

Арифметические инструкции, первая программа, знаковое и беззнаковое переполнение, регистр флагов, изменение потока управления программы с учетом регистра флагов. Отображение арифметических команд языка С на язык ассемблер.

В.И. Юров [3]. Bryant, O'Hallaron Computer Systems. A programmer perspective [1]

Тема 3. Архитектура ЭВМ, структура ассемблерной программы

Архитектура фон-Неймана. Гарвардская архитектура. Их различия, сильные и слабые стороны. Конвейер инструкций.

Bryant, O'Hallaron Computer Systems. A programmer perspective [1]. A. Tanenbaum. Structured Computer Organization [5]

Тема 4. Реализация управляющих операторов языка Си: условная передача данных, организация циклов.

Инструкции перехода. Организация условных и безусловных переходов. Организация циклов. Отображения конструкций управления потока языка С в язык ассемблер.

Bryant, O'Hallaron Computer Systems. A programmer perspective [1]

Тема 5. Размещение указателей, массивов, матриц и структур в памяти.

Работа с памятью и с указателями на уровне ассемблера. Размещение массивов в памяти. Доступ к элементам массива на уровне ассемблера. Организация доступа к полям структуры, выравнивание структур. Отображения с языка С.

Bryant, O'Hallaron Computer Systems [1].

Тема 6. Этапы компиляции. Размещение программы в памяти. Структура файлов ELF.

Этапы компиляции программ. Промежуточное представление программ. Оптимизирующие компиляторы. Генерация кода. Распределение регистров. Статическая компоновка программ. Разбор формата объектных файлов ELF. Создание статической библиотеки. Положительные

и отрицательные стороны статических библиотек. Создание динамических библиотек. Размещение динамических библиотек в памяти. Компоновка программ с динамическими библиотеками. Код, не зависящий от адреса (PIC)

Bryant, O'Hallaron Computer Systems [1]

Тема 7. Стек и куча

Организация стека и кучи в адресном пространстве процесса. Принципы размещения данных в стеке и динамического выделения памяти в куче. Изменение указателя стека, время жизни объектов, выделение и освобождение динамической памяти, основные ошибки управления памятью.

Bryant, O'Hallaron Computer Systems [1]

Тема 8. Организация вызовов функций, стековый кадр функции. Оптимизация хвостового вызова. Вызов внешних функций.

Организация вызовов функций. Соглашения о вызовах. Передача аргументов. Адрес возврата. Фрейм функции в стеке. Возврат значения. Пролог и эпилог функций.

Bryant, O'Hallaron Computer Systems [1]

Тема 9. Числа с фиксированной и плавающей запятой.

Числа с фиксированной запятой. Числа с плавающей запятой. Представление вещественных чисел по стандарту IEEE-754. Основные арифметические операции над двоичным представлением вещественных чисел.

В.И. Юров. Ассемблер [3]

Тема 10. Аппаратура ЭВМ. Физические основы, логические вентили, полусумматор, полный сумматор, АЛУ.

Принцип работы транзисторов. Логические вентили (not-and, not-or, and, or, not). Построение логических схем. Схема XOR. Построение полусумматора, полного сумматора. Мультиплексоры и декодеры. Построение АЛУ.

Bryant, O'Hallaron Computer Systems [1]

A. Tanenbaum. Structured Computer Organization [5]

Тема 11. Статическая память. SR-защелка. D-защелка. D-триггер. Регистр.

Физические основы построения бита памяти. SR-защелка. D-защелка. D-триггер. Регистр. Устройство управления (Control Unit).

A. Tanenbaum. Structured Computer Organization [5]

2.3.3. Краткое содержание семинарских/практических занятий/лабораторного практикума

Составление программ на языке ассемблера.
Прямые задачи: Написать эквивалентную данному фрагменту на языке Си программу на языке ассемблера (для разных конструкций языка Си).
Обратные задачи: восстановить фрагмент Си кода из данной ассемблер-программы.
Организация вызовов функций с использованием разных соглашений о вызовах.
Работа в среде SASM.

2.3.4. Материально-техническое обеспечение дисциплины

Компьютеры с доступом к интернету
Среда программирования SASM
Платформа e-judge

2.4. Модульная структура дисциплины с распределением весов по формам контролей

Формы контролей	**Вес формы (форм) текущего контроля в результирующей оценке текущего контроля (по модулям)**		**Вес формы промежуточного контроля в итоговой оценке промежуточного контроля**		**Вес итоговой оценки промежуточного контроля в результирующей оценке промежуточных контролей**		**Вес итоговой оценки промежуточного контроля в результирующей оценке промежуточных контролей (семестровой оценке)**	**Веса результирующей оценки промежуточных контролей и оценки итогового контроля в результирующей оценке итогового контроля**
Вид учебной работы/контроля	**M1**[1]	**M2**	**M1**	**M2**	**M1**	**M2**		
Контрольная работа *(при наличии)*	0.5	0.5						
Устный опрос *(при наличии)*								
Тест *(при наличии)*								
Лабораторные работы *(при*								

[1] Учебный Модуль

наличии)								
Письменные домашние задания *(при наличии)*								
Реферат *(при наличии)*								
Эссе *(при наличии)*								
Проект *(при наличии)*								
Другие формы (при наличии)								
Веса результирующих оценок текущих контролей в итоговых оценках промежуточных контролей						0.5		
Веса оценок промежуточных контролей в итоговых оценках промежуточных контролей						0.5		
Вес итоговой оценки 1-го промежуточного контроля в результирующей оценке промежуточных контролей							0.5	
Вес итоговой оценки 2-го промежуточного контроля в результирующей оценке промежуточных контролей							0.5	
Вес результирующей оценки промежуточных контролей в результирующей оценке итогового контроля								0.5
Вес итогового контроля (Экзамен/зачет) в результирующей оценке итогового контроля								0.5
	∑ = 1	**∑ = 1**	**∑ = 1**	**∑ = 1**	**∑ = 1**	**∑ = 1**	**∑ = 1**	**∑ = 1**

3. Теоретический блок

3.1. Материалы по теоретической части курса

3.1.1. Учебник(и);

1. Randal E. Bryant, David R. O'Hallaron. Computer Systems: A Programmer's Perspective. Prentice Hall.
2. Рендел Брайант, Давид О'Халларон. Компьютерные системы: Архитектура и программирование. Взгляд программиста. Пер. с англ., СПб.: БХВ, 2005.
3. В.И. Юров Assembler. Учебник для вузов. 2-е изд., СПб.: Питер, 2003.

4. С.В. Зубков. Assembler для DOS, Windows, и Unix. Язык низкого уровня. М., ДМК, 2000 г.

5. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е изд. - СПб.: Питер, 2013.

3.1.2. Учебное(ые) пособие(я);

6. Е.А. Кузьменкова, В.С. Махнычев, В.А. Падарян. Семинары по курсу «Архитектура ЭВМ и язык ассемблера»: учебно-методическое пособие. Часть 1. Издание 2-е, дополненное. М., Изд-во МГУ, МАКС Пресс, 2014. – 80 с.

7. Е.А. Кузьменкова, В.А. Падарян, М.А. Соловьев Семинары по курсу «Архитектура ЭВМ и язык ассемблера»: учебно-методическое пособие. Часть 2. М., изд-во МГУ, МАКС Пресс, 2014. – 100 с.

3.1.3. Курс лекций;

3.1.4. Краткие конспекты лекций;

3.1.5. Электронные материалы (электронные учебники, учебные пособия, курсы и краткие конспекты лекций, презентации PPT и т.п.);

3.1.6. Глоссарий/терминологический словарь;

3.1.7. др. варианты материалов, необходимых для освоения учебной программы дисциплины.

4. Фонды оценочных средств

4.1. Планы практических и семинарских занятий

1. Двоичное представление чисел, шестнадцатеричное представление чисел. Типы word, double word, quad word. Инструкция mov. Арифметические и логические инструкции. Организация ввода и вывода в среде SASM.

2. Инструкции деления и умножения.

3. Инструкции пересылки с расширением, инструкции расширения, секции data, bss. Чтение из памяти.

4. Логические и арифметические сдвиги. Регистр флагов. Инструкции условной пересылки.

5. Инструкция сравнения. Реализация условных конструкций языка С в ассемблере.

6. Реализация циклов в ассемблере. Инструкция loop.

7. Указатели, массивы, матрицы и структуры.

8. Локальные массивы и их размещение в стеке.

9. Передача аргументов и возврат значений. Соглашение cdecl.

10. Вложенные вызовы функций. Сохранение регистров.

11. Вызов внешних функций из ассемблерной программы.

12. Динамическое выделение памяти, работа с кучей.

4.2. Планы лабораторных работ и практикумов

4.3. Материалы по практической части курса

4.3.1. Учебно-методические пособия;

1. Е.А. Кузьменкова, В.С. Махнычев, В.А. Падарян. Семинары по курсу «Архитектура ЭВМ и язык ассемблера»: учебно-методическое пособие. Часть 1. Издание 2-е, дополненное. М., Изд-во МГУ, МАКС Пресс, 2014. – 80 с.

2. Е.А. Кузьменкова, В.А. Падарян, М.А. Соловьев Семинары по курсу «Архитектура ЭВМ и язык ассемблера»: учебно-методическое пособие. Часть 2. М., изд-во МГУ, МАКС Пресс, 2014. – 100 с.

4.3.2. Учебные справочники;

4.3.3. Задачники (практикумы);

Набор практических задач по языку ассемблера в системе e-judge.

4.3.4. Наглядно-иллюстративные материалы;

Презентации к лекциям, схемы архитектуры процессора, схемы логических элементов и устройств памяти.

4.3.5. др. виды материалов.

4.4. Вопросы и задания для самостоятельной работы студентов

Задачи для самостоятельного выполнения в среде e-judge (15 задач к первому модулю, 19 задач ко второму модулю)

4.5. Тематика рефератов, эссе и других форм самостоятельных работ

4.6. Образцы вариантов контрольных работ, тестов и/или других форм текущих и промежуточных контролей

Модуль 1.

1. Вычислить значения регистра al и флагов OF, SF, CF, ZF

```
MOV AL, 130
ADD AL, 126
```

2. Написать фрагмент кода на языке ассемблера, вычисляющий значение переменной a. Напечатать результат на экран. На стандартном потоке ввода задаются пять целых чисел через пробел: a, b, c, d и x. Если вторая половина выражения равна нулю, поставить в “a” значение 0.

```
static int a, b;
static short c, d, x;
a =(a*c+d)/(d*c+(c&3)+c%x)
```

3. Напечатать на экран десятичное число, представляющее первый и предпоследний биты заданной 32-битной переменной. Переменная вводится пользователем.

4. Напишите программу, которая выводит числа от **A** до **B**. Но: Если число **делится на 3**, вместо него выведите **"Fizz"**. Если число **делится на 5**, вместо него выведите **"Buzz"**. Если число **делится и на 3, и на 5**, выведите **"FizzBuzz"**. В остальных случаях просто выведите само число.

5. Дан ассемблерный код, реализующий код на языке Си.

```
        xor cx, cx
l1:
        cmp cx, [a]
        jae l2
        mov eax, [b]
        movzx ebx, cx
        inc ebx
        mul ebx
        mov [b], eax
        inc cx
        jmp l1
l2:
```

Требуется восстановить пропуски в Си-коде.

```
static _____ a;
static _____ b;
```

```
for (_____) {
  ____
}
```

Модуль 2.

1. Для приведенного фрагмента Си-кода требуется написать соответствующий фрагмент ассемблерной программы.

```
static char* a;
static char** b;
char c = (*a) - **(b)
```

2. На стандартном потоке ввода задается натуральное число N (N > 0), после которого следует последовательность из N целых чисел. На стандартный поток вывода напечатайте последнее нечетное число. Гарантируется, что последовательность содержит хотя бы одно нечетное число.

3. На стандартный поток ввода задаются некоторые беззнаковые целые числа. Последовательность заканчивается нулем, который в нее не входит. Нужно вывести каждое число, полученное из функции описанной далее. Требуется использовать функцию int is_binary_palindrome(unsigned int n), которая как аргумент получает беззнаковое целое число и возвращает 1 если число является палиндромом в двоичном представлении, и 0 в обратном случае.

4. После компиляции функции из С-кода получается следующий ассемблер.

```
section .text
f:
   push ebp
   mov ebp, esp
   sub esp, 10
```

```
    xor ecx, ecx
s:
    cmp ecx, 10
    je e
    lea esi, [ebp+10]
A:
    mov ax, [esi+ecx]
    cmp ax, 37
    jg B
    mov [esp+ecx], ax
    jmp C
B:
    mov ax, [esp+ecx]
    mov [esi+ecx], ax
C:
    add ecx, 2
    jmp s
e:
    cwde
    mov esp, ebp
    pop ebp
    ret
```

Требуется восстановить пропуски в Си-коде.

```
struct S {
  signed char c1;
  signed char c2;
  _____ a[__];
};

int f(struct S x) {
```

```
_____ arr[__];
int i;
for (_________) {
        if (_____) {
            ______;
        } else {
            ______;
        }
}
return ______;
}
```

4.7. Перечень экзаменационных вопросов

1. Регистры архитектуры x86. Регистры общего назначения. mov vs movsx vs movzx. Арифметические и логические инструкции. Инструкции конвертации.
2. Регистр флагов. Инструкции adc и sbb. Инструкция cmp и test. Условный mov, set.
3. Инструкции условного и безусловного перехода. Реализация условных конструкций и циклов C.
4. Архитектура фон Неймана. Гарвардская архитектура. Цикл инструкций (instruction cycle: fetch/decode/execute). Конвейер инструкций (instruction pipelining). Узкое место фон Неймана (von-Neumann Bottleneck).
5. Порядок байтов (endianness). Указатели. Массивы.
6. Матрицы. Структуры, выравнивание (alignment), padding.
7. Этапы компиляции (компоновка вкратце).
8. Статическая компоновка (static linking). Loading и динамическая компоновка (dynamic linking).
9. Стек (stack) и куча (heap). Регистры esp и ebp. Инструкции управления стеком.
10. Процедуры. Вызов и возврат из процедуры. Соглашения о вызове (calling conventions).
11. Структура ассемблерной программы. Структура файлов ELF. Вызов внешних функций (external functions).

12. Хвостовой вызов. Хвостовая рекурсия. Оптимизация хвостового вызова. Оптимизация хвостовой рекурсии.

13. Числа с фиксированной запятой (fixed point numbers). Числа с плавающей запятой (floating point numbers). Суммирование и умножение.

14. Транзисторы. NAND. Базовые логические вентили (not, and, or, xor, nor). Полусумматор (half adder), полный сумматор (full adder). Реализация вычитания при помощи сумматора. Мультиплексор. АЛУ.

15. SR-защелка (SR-Latch). D-защелка (D-Latch). Data flip-flop. Регистр. Чтение и запись в регистр. Декодер. Устройство управления (Control Unit).

4.8. Образцы экзаменационных билетов

1. Регистры архитектуры x86. Регистры общего назначения. mov vs movsx vs movzx. Арифметические и логические инструкции. Инструкции конвертации.

2. Матрицы. Структуры, выравнивание (alignment), padding.

3. Структура ассемблерной программы. Структура файлов ELF. Вызов внешних функций (external functions).

4. Задача.

4.9. Образцы экзаменационных практических заданий

Написать функцию для суммирования двух 8 байтовых чисел.

4.10. Банк тестовых заданий для самоконтроля

Кодирование чисел: Представить следующие числа в прямом, обратном и дополнительном коде: 8, 9, -9, -12

Число	прямой	обратный	дополнительный
8	00001000	00001000	00001000
9	00001001	00001001	00001001
-9	10001001	11110110	11110111
-12	10001100	11110011	11110100

Система команд: Пересылка данных (mov, movsx, movzx). Определить значение регистра после операций:

- mov al, 10
- mov al, 127
- mov bh, 300 // bh = 44, bx = 11264
- mov ax, 0xfffff //ax = 0xffff

mov al, 1 // ax =0xff01, al = 1, ah = 0xff

- mov bx, 45

  movzx ax, bl

- mov bx, -45 // bx = 0xffd3

  movzx ax, bl // ax = 0xd3

- mov bl, -1 // bl = 0xff, bx = 0xff

  movsx ax, bl // ax = 0xffff

4.11. Методики решения и ответы к образцам тестовых заданий

При выполнении заданий необходимо учитывать разрядность регистра. При записи значения, не помещающегося в регистр, сохраняются младшие биты значения. Инструкция movzx выполняет расширение нулями, movsx — знаковое расширение.

Ответы к заданию «Кодирование чисел» (8-битное представление): 8 — прямой 00001000, обратный 00001000, дополнительный 00001000; 9 — прямой 00001001, обратный 00001001, дополнительный 00001001; -9 — прямой 10001001, обратный 11110110, дополнительный 11110111; -12 — прямой 10001100, обратный 11110011, дополнительный 11110100.

Ответы к заданию «Пересылка данных»: mov al, 10 → AL = 10 (0x0A); mov al, 127 → AL = 127 (0x7F); mov bh, 300 → BH = 44 (0x2C), BX = 11264 (0x2C00); mov ax, 0xfffff → AX = 0xFFFF, затем mov al, 1 → AX = 0xFF01; mov bx, 45; movzx ax, bl → AX = 45 (0x002D); mov bx, -45; movzx ax, bl → AX = 211 (0x00D3); mov bl, -1; movsx ax, bl → AX = 0xFFFF (-1 при знаковой интерпретации).

5. Методика оценивания

5.1. Распределение баллов между видами учебной деятельности и контрольными мероприятиями;

В каждом учебном модуле текущий контроль формируется из результатов контрольной работы (60 %) и выполнения домашних заданий в системе e-judge (40 %). Итоговая оценка каждого промежуточного контроля формируется из результирующей оценки текущего контроля (50 %) и оценки промежуточного контроля (50 %).

5.2. Критерии оценивания по каждому оценочному средству;

Контрольные работы и практические задания оцениваются по правильности полученного результата, корректности реализации алгоритма и соблюдению требований задания. Задания, выполняемые в системе e-judge, оцениваются по результатам прохождения

автоматических тестов. При оценивании теоретической части экзамена учитываются полнота, правильность и обоснованность ответа.

5.3. Шкалу перевода баллов в итоговую оценку;

5.4. Порядок формирования итоговой оценки по дисциплине;

Результирующая оценка промежуточных контролей определяется как среднее взвешенное оценок первого и второго модулей с равными весами 0,5. Итоговая оценка по дисциплине формируется из результирующей оценки промежуточных контролей с весом 0,5 и оценки экзамена с весом 0,5.

5.5. Минимальные требования для получения положительной оценки;

5.6. Соответствие оценочных средств результатам обучения дисциплины.

Теоретические вопросы и экзамен — проверка знаний.
Практические задания и контрольные работы — проверка умений применять знания при анализе и написании ассемблерных программ.
Задания e-judge — проверка практических навыков программирования и анализа низкоуровневого кода.

6. Методический блок

6.1. Методика преподавания

Преподавание учебной дисциплины «Архитектура ЭВМ и язык ассемблера» строится на сочетании лекций, практических занятий и различных форм самостоятельной работы студентов.

На лекциях рассматриваются основные вопросы по темам, составляющим содержание дисциплины, с акцентом на наиболее сложные проблемы.

На практических занятиях делается акцент на проверке теоретических знаний и их применении при решении практических задач и программировании на языке NASM в среде программирования SASM.

Рабочей программой дисциплины предусмотрена самостоятельная работа студентов, включающая работу с литературными и электронными ресурсами, выполнение домашних заданий в среде e-judge.

Промежуточный контроль проводится с использованием платформы e-judge.

6.1.1. Методические рекомендации для студентов по подготовке к семинарским, практическим или лабораторным занятиям, по организации самостоятельной работы студентов при изучении конкретной дисциплины.

Для подготовки к практическим занятиям студентам рекомендуется предварительно ознакомиться с соответствующим теоретическим материалом и примерами программ. Самостоятельная работа включает изучение рекомендованной литературы, повторение материалов лекций и выполнение практических заданий в системе e-judge. При подготовке к контрольным мероприятиям рекомендуется повторить соответствующие темы курса и самостоятельно решить предложенные практические задачи.