Вопросы краткие по главам

Вопросы по главам

ГЛАВА 1

1. **Что включает в себя понятие архитектуры компьютера?**
   принципы, на которых основана работа компьютера, а также проблемы, связанные с созданием практически используемых компонентов
2. **Какие простые (основные) типы существуют?**
   целые числа, числа с плавающей точкой, символы и логические значения.
3. **Какие основные элементы присутствуют в процедурных языках программирования?**
   переменные, значения элементарных типов и операции (арифметические, логические и отношения между значениями.)
4. **Что включает в себя модель компьютера?**
   процессор, память, устройства ввода-вывода, систему адресации
5. **Что характеризует память?**
   память состоит из набора локаций, каждая из которых имеет уникальный адрес и может содержать значение элементарного типа. *Текст также упоминает о том, что локации могут иметь символические метки, которые используются вместо адресов, и что память может быть организована в виде массивов или структур данных.*
6. **Как можно интерпретировать содержимое ячейки памяти?**
   содержимое ячейки памяти может быть интерпретировано как значение элементарного типа, которое было сохранено в этой ячейке. Каждая ячейка имеет уникальный адрес, который может быть использован для доступа к ее содержимому. Текст также упоминает о том, что ячейки могут иметь символические метки, которые могут использоваться вместо адресов.
7. **Что характеризует процессор?**
   самый главный отвечает за выполнение машинных команд. может иметь несколько ядер и выполнять несколько команд одновременно, описывает различные типы команд, которые могут быть выполнены процессором, включая арифметические, логические и операции с памятью.

8. **Какие виды адресации существуют?**
   непосредственную, прямую, регистровую, косвенную и индексную адресацию.
9. **Что входит в состав машинной команды?**
   операция и операнды
10. **Что входит в состав ассемблерной команды?**
    мнемоники операции , операндов и `комментариев`
11. **Какую задачу выполняет ассемблер?**
    Задача ассемблера - преобразовать ассемблерный код в машинный код ассемблер выполняет функцию трансляции между ассемблерным и машинным языками.
12. **Какую задачу выполняет компилятор?**
    преобразовать исходный код программы, написанный на высокоуровневом языке программирования (например, на языке C, Java, Python), в машинный код (или ассемблерный код), который может быть выполнен процессором. Таким образом, компилятор выполняет функцию трансляции между высокоуровневым и машинным (или ассемблерными) языками.
13. **Что обеспечивает соответствие цифр двоичной системы счисления и логических значений?**
    возможность описания арифметических операций для двоичной системы с помощью логических функций, если двоичные цифры интерпретируются как логические значения
14. **Какая логическая функция описывает сумму двух бинарных цифр?**
    XOR
15. **Что определяет значение функции переключения?**
    состояние переключателя, то есть значение на выходе переключателя, которое определяет уровень сигнала на управляющем выводе транзистора.
16. **На что влияет значение переключателя аргумента?**
    состояние переключателя.
17. **Что обеспечивают последовательные схемы?**
    память и способность обрабатывать последовательности входных сигналов.
18. **Что обеспечивают комбинационные схемы?**
    Комбинационные схемы - это схемы, которые выполняют логические

операции над входными данными и выдают результат на выходе в соответствии с заданной логикой.

ГЛАВА 2

1. **Что обеспечивает представление комплемента для знаковых чисел? (ДОПОЛНЕНИЕ)**
   обеспечивает упрощение операций сложения и вычитания, а также представление отрицательных чисел в компьютерных системах
2. **Как представление комплемента упрощает работу процессора?**
   позволяет выполнять арифметические операции над знаковыми числами так же, как и над беззнаковыми числами. *вычитание == сложение*
3. **Когда (не) игнорируется перенос с наиболее значимой позиции при арифметике целых чисел? XXXXXXX**
   игнорируется:
   При использовании беззнаковых целых чисел.
   При использовании представления комплемента для знаковых целых чисел.

   значит блять не игнорируетс я в остальных случаях, он высрал какую то хуйню
4. **Что указывает на выход за пределы диапазона?**
   Это означает, что результат операции не может быть представлен в данной системе представления чисел.
5. **Какие части включает в себя машинная нормализованная форма?**
   Знак, экспоненту и мантиссу (дробную часть числа). Экспонента определяет порядок числа, а мантисса определяет его значащие цифры.
6. **Зачем введена константа настройки (как она используется)?**
   Константа настройки введена в машинной нормализованной форме (МНФ) для того, чтобы расширить диапазон представления чисел и улучшить точность представления.
   Для того, чтобы расширить диапазон значений экспоненты, в МНФ используется константа настройки, которая добавляется к значению экспоненты.
7. **Как выполняется арифметика чисел, представленных машинной нормализованной формой**

В машинной нормализованной форме (МНФ) арифметика выполняется путем сначала выравнивания экспонент, затем сложения или вычитания мантисс и, при необходимости, нормализации результата.

8. **Где происходит выход за пределы диапазона в арифметике машинной нормализованной формы (в экспоненте или дробной части)?**
   в экспонент
9. **Где отбрасываются лишние цифры в арифметике машинной нормаизованной формы (в экспоненте или дробной части)?**
   в дробной части
10. **Как выполняются преобразования чисел из десятичной в двоичную систему счисления (и наоборот)?**

- Разделить исходное число на 2.
- Записать остаток от деления (0 или 1) в качестве младшего разряда двоичного числа.
- Если результат деления больше 0, то повторить шаги 1-2, используя результат деления в качестве исходного числа.
- Если результат деления равен 0, то запись двоичного числа завершена.

11. **Какое двоичное число получается после преобразования десятичного числа 4,25?**
100,01

12. **Какое десятичное число получается после преобразования двоичного числа 100.01?**
Отв 4,25

13. **Какой шестнадцатеричный код получается после преобразования двоичного числа 11010?**
Отв 1A

14. **Какое двоичное число получается после преобразования шестнадцатеричного числа 1A?**
Отв 11010

15. **Каково представление комплемента 2 для двоичного числа -100?**
-ответ говна ответ 1100
-ответ говна ответ 1100-7+52 000 000

16. **Происходит ли выход за пределы диапазона при сложении двоичных чисел 01000100 и 01110000 (для обеих интерпретаций этих чисел)?**
для знакового происходит для беззнакового нет - фактиш

17. **На основе чего определяется действие отношений между (не)знаковыми числами?**
Отношения между (не)знаковыми числами определяются на основе их двоичного представления и алгоритма сравнения двоичных чисел.

18. **Какова машинная нормализованная форма (MNF8) двоичного числа -101?**
Для получения машинной нормализованной формы (MNF8) двоичного числа -101 необходимо выполнить следующие шаги:
1) -1 | 1.01 * 2^2
2) -1→1|10 + 1000|010(потому что 3 бита остается для мантиссы)
3) 11010010MNF8
19. **Какому двоичному числу соответствует машинная нормализованная форма 11010010MNF8?**
1) 1→-1 |1010 - 1000→10 - показатель экспоненты |010(мантисса)
2) -1 * 10^10(2) * 1.010
3) -101

ГЛАВА 3

1. Какие существуют языки низкого и высокого уровня программирования?
   ОТВ от гпт: Существуют языки программирования низкого уровня, такие как ассемблер, и языки высокого уровня, такие как C++, Python, и Java.
2. **В чем разница между ассемблерным и машинным языками?**
   Ассемблерный язык - это символическое представление машинного языка, в котором используются метки и символические обозначения для операций и операндов (eng. Labels).

3. **Когда оправдано использование ассемблерного и машинного языков?**
   когда не существует трансляторов для процедурных языков программирования (компиляторов) или когда такие трансляторы создаются, а также в случаях, когда необходимо использовать возможности компьютера на пределе его возможностей
4. **Какой адресный пространство предоставляют 2 бита?**
   2 бита могут представлять 4 различных бинарных числа. Если каждое из них представляет адрес только одного места в памяти, то общее адресное пространство, которое может быть представлено 2 битами, составляет ***4 адреса***.
5. **Какие команды поддерживают арифметику с многократной точностью?**
   FADD (сложение), FSUB (вычитание), FMUL (умножение), FDIV (деление)
6. **Какие команды в процессе выполнения не изменяют условные биты?**
   JUMPS - JE JG JNE….
   NEG
   NOP
7. **Какие арифметические команды существуют?**
   ADD SUB MUL DIV INC DEC NEG и т.д.
8. **Какие команды для работы с битами существуют?**
   AND (логическое И)
   OR (логическое ИЛИ)
   XOR (логическое исключающее ИЛИ)
   NOT (логическое отрицание)
   TEST (логическое тестирование)
   SHL (сдвиг влево)
   SHR (сдвиг вправо)
   ROL (циклический сдвиг влево),
   ROR (циклический сдвиг вправо) и т.д.
9. **Какие управляющие команды существуют?**
   1) Jcc (Jump if Condition is met): JE (Jump if Equal), JG, JNG, JL, JA …
   2) JMP (Jump)
   3) CALL (Call Procedure) и RET (Return from Procedure)
   4) INT (Interrupt), INTO (Interrupt on Overflow)
   5) PUSH (Push Word or Doubleword Onto the Stack), POP (Pop a Value from the Stack)

6) IN (Input from Port), OUT (Output to Port)
7) CLC (Clear Carry Flag), STC (Set Carry Flag), CLD (Clear Direction Flag), STD (Set Direction Flag) …

10. **Какие комбинации операндов используют команды перехода?**
    JMP - LABEL
    Jcc - LABEL (FLAGS)

11. **Какие команды позволяют обнаруживать выход за пределы диапазона?**
    JO, JC

12. **Какие команды имеют два операнда?**
    1) *Арифметические:* ADD (Add) ,SUB (Subtract), MUL (Multiply), DIV (Divide) и т.д.,
    2) *Логические:* AND (Logical AND), OR (Logical OR), XOR (Exclusive OR) и т.д.
    3) *Сравнения:* CMP (Compare), TEST (Logical Compare), CMPS (Compare String)
    4) *Перемещения\копирования:* MOV (Move), XCHG (Exchange), MOVS(B/W/L) (Move String), LEA(Load Effective Address)

13. **Какие команды имеют один операнд?**
    1) *Арифметические:* INC (Increment), DEC (Decrement), NEG (Negate) и т.д
    2) *Логические:* NOT (Logical NOT), SHL (Shift Left), SHR (Shift Right) и т.д.
    3) *Работа со стеком:* PUSH, POP
    4) MOV $IMM, MEM/REG
    5) *Ветвления:* JMP, Jcc. CALL, RET
    6) *Прерывания:* INT (Interrupt),

14. **Какие команды не имеют операндов?**
    NOP (No Operation):
    HLT (Halt):
    CLC (Clear Carry Flag):
    STC (Set Carry Flag):
    CMC (Complement Carry Flag):
    CLD (Clear Direction Flag):
    STD (Set Direction Flag):
    CLI (Clear Interrupt Flag):
    STI (Set Interrupt Flag):
    RET (Return):

NOPW (No Operation Word)
INTO

15. **Какие ассемблерские директивы существуют?**
    EQU (Equate) и SET (Set)
    ALIGN (Align) и ORG (Origin)
    DB (Define Byte), DW (Define Word), DD (Define Doubleword) и DQ (Define Quadword),
    SEGMENT (Segment) и ENDS (End Segment),
    INCLUDE (Include) и MACRO (Macro),
    TITLE (Title), SUBTITLE (Subtitle) и COMMENT (Comment)
    LIST (List), NOLIST (No List) и LISTMACRO (List Macro),
    EXTERN (External), PUBLIC (Public) и GLOBAL (Global),
    IF (If), ELSE (Else) и ENDIF (End If)
    ERROR (Error) и WARNING (Warning)
16. **Что характеризует подпрограмму?**
    подпрограмма - это фрагмент программного кода, который может быть вызван из основной программы для выполнения определенной задачи.
17. **Что характеризует макро?**
    Макро - это фрагмент кода, который может быть вызван из другого кода и заменен на определенный набор инструкций.
18. **Что характеризует функцию?**
    Функция - это фрагмент кода, который выполняет определенную задачу и возвращает результат. Она может быть вызвана из другого кода и может принимать аргументы. Функции обычно используются для разделения кода на более мелкие и управляемые блоки, что упрощает понимание и сопровождение кода.
19. **Что характеризует процедуру?**
    Процедура - это фрагмент кода, который выполняет определенную задачу, но не возвращает результат.
20. **Какие ассемблерские команды вводятся для ассемблерных подпрограмм?**
    Для ввода ассемблерных подпрограмм используются две директивы: PROC и ENDP.
21. **Что связано с вызовом ассемблерной подпрограммы?**
    При вызове ассемблерной подпрограммы необходимо передать ей аргументы, если они требуются.

22. **Что связано с вызовом макро?**
    При вызове макро необходимо указать его имя и передать ему аргументы, если они требуются.

23. **Как использование подпрограмм влияет на длину программы и время ее выполнения?**
    Влияет странно, может и увеличеть время выполнения, а может и уменьшить, зависит от того на сколько эффективно мы будем использовать подпрограмм

24. **Как использование макросов влияет на длину программы и время ее выполнения?**
    Влияет странно, может и увеличеть время выполнения, а может и уменьшить, зависит от того на сколько эффективно мы будем использовать макросы

25. **Кто выполняет замену вызова макроса модифицированными командами из его определения?**
    выполняет макро-препроцессор.macro preprocessor

26. **Где и когда определены глобальные переменные?**
    Глобальные переменные определены вне тела любой функции или процедуры, обычно в начале программы.

27. **Где и когда определены локальные переменные?**
    Локальные переменные определены внутри тела функции или процедуры и существуют только во время выполнения этой функции или процедуры.
    Чаще всего эти переменные выделяются на стеке

28. **Что размещается в стеке?**
    В стеке размещаются локальные переменные функций и процедур, а также адреса возврата, сохраняемые при вызове функций и процедур.

29. **Как управлять стеком?**
    управление стеком осуществляется с помощью специальных инструкций процессора (PUSH, POP), которые позволяют заносить и извлекать данные из стека. и стек поинтер

30. **Что содержит фрейм?**
    фрейм (stack frame) - это набор локаций на стеке, который содержит информацию о вызове функции или процедуры. Фрейм обычно содержит следующие элементы:
    Адрес возврата — ВСЕГДА! ОСТАЛЬНОЕ ОПЦИОНАЛЬНО

Аргументы (если есть)
Регистры (сохранены calle функцией по СС)
Указатель на предыдущий фрейм (сохраняется esp, и объявляетмя новый ebp для текущей функции\процедуры)
Другие элементы - в зависимости от конкретной реализации и используемых компиляторов, фрейм может содержать дополнительные элементы, такие как сохраненные значения регистров, информацию о типах данных и т.д.