汇编语言课程简答题题库（第2章-第9章）

> 题目风格参照课程 PPT 中的概念、比较、过程说明和论证性表述，适合按 PPT 原句整理答案。

第二章 80X86计算机组织

1. 请简述以存储器为中心的计算机结构由哪些主要部件构成。
2. 请说明面向总线的体系结构中，分散连接和总线连接的区别。
3. 请分别说明数据总线、地址总线和控制总线的作用。
4. 请说明内部数据总线宽度和外部数据总线宽度的含义。
5. 请简述 80X86 微处理器系列的发展特点。
6. 请说明高级语言和汇编语言在 CPU 兼容性方面的区别。
7. 请说明字长增加对计算机性能有什么影响。
8. 请说明地址总线宽度与寻址空间之间的关系。
9. 请简述 80286 开始增加的保护模式提供了哪些功能。
10. 请简述 80386 开始增加的虚拟 8086 工作模式的作用。
11. 请说明 80486 开始把协处理器集成到 CPU 芯片中的意义。
12. 请简述 80X86 处理器实模式、保护模式和虚拟 8086 模式之间的关系。
13. 请说明 CPU 复位或加电后处理器首先工作在哪一种模式。
14. 请说明实模式下 80X86 内存寻址方式的基本特点。
15. 请说明实模式下为什么所有段都可以读、写和执行。
16. 请简述保护模式下 CPU 提供的主要功能。
17. 请说明实模式下应用程序为什么可能破坏操作系统和其他应用程序。
18. 请说明保护模式如何通过特权级和内存分段分页限制应用程序行为。
19. 请简述保护模式下段大小、分页、虚拟内存和多任务支持的特点。
20. 请简述虚拟 8086 模式的基本特点。
21. 请说明 ARMv8-A 中异常级别 EL0、EL1、EL2、EL3 通常分别对应什么软件层次。
22. 请说明 ARMv8-A 的两个安全状态分别是什么。
23. 请说明 ARMv8-A 的 AArch32 和 AArch64 两种执行状态的区别。
24. 请列举 ARM 异常事件的主要类型。
25. 请说明基于微处理器的计算机系统由哪些部分构成。
26. 请说明外设接口寄存器通常包括哪三类寄存器及其作用。
27. 请说明外设接口寄存器的两种编址方式。
28. 请说明 8086 中通用寄存器、段寄存器、指针和变址寄存器的基本用途。
29. 请说明标志寄存器 FLAGS 中状态标志和控制标志的作用区别。
30. 请说明 CF、ZF、SF、OF 标志位的含义。
31. 请说明 8086 存储器为什么采用分段管理。
32. 请说明实模式下逻辑地址、段地址、偏移地址和物理地址的关系。
33. 请写出实模式下物理地址的计算公式。
34. 请说明 CS、DS、SS、ES 四个段寄存器的用途。
35. 请说明段寄存器与偏移地址的缺省组合规则。
36. 请说明汇编指令中给出逻辑地址后，机器如何得到物理地址。
37. 请说明各段之间可以存在相邻、相离和重叠关系的原因。
38. 请简述引出保护模式存储器寻址的主要原因。
39. 请说明保护模式下逻辑地址由哪两部分组成。

40. 请说明保护模式下选择器为什么不能直接表示段基地址。
41. 请说明保护模式下存储器寻址的基本过程。
42. 请说明实模式和保护模式下存储器寻址过程的区别。
43. 请说明描述符的长度、组成和用途。
44. 请说明描述符中段基地址、界限、访问权限和附加字段的作用。
45. 请说明选择器的 INDEX、TI、RPL 字段的含义。
46. 请说明 GDT、LDT、IDT 三类描述符表的作用。
47. 请说明程序不可见寄存器的含义，并举例说明。
48. 请说明为什么保护模式下需要描述符高速缓冲存储器。
49. 请说明大端格式和小端格式的区别，并指出 X86 采用哪一种。
50. 请说明 ARM 中 MMU 的作用。

第三章 指令系统和寻址方式

1. 请说明什么是指令系统。
2. 请说明指令系统为什么决定了计算机能执行的全部基本操作。
3. 请说明常见指令格式中零地址、一地址和二地址指令的区别。
4. 请说明学习指令系统的两个关键点。
5. 请说明什么是寻址方式。
6. 请说明操作数通常可以保存在哪些位置。
7. 请说明机器指令格式 OP dst, src 中各部分含义。
8. 请说明立即寻址方式的特点、用途和限制。
9. 请说明寄存器寻址方式的特点。
10. 请说明为什么立即寻址方式只能用于源操作数字段。
11. 请说明除立即寻址和寄存器寻址外，其他数据寻址方式如何取得操作数。
12. 请说明操作数的段基地址在实模式和保护模式下分别如何获得。
13. 请说明指令中操作数逻辑地址的一般格式。
14. 请说明段寄存器名可以缺省和特别指定的含义。
15. 请列出指令、堆栈操作、普通变量、字符串源串、字符串目标串和 BP 作基址时的约定段寄存器。
16. 请说明什么是有效地址 EA。
17. 请说明有效地址 EA 的四种成分。
18. 请说明基址、变址、比例因子和位移量分别有什么作用。
19. 请写出有效地址 EA 的一般计算公式。
20. 请说明 16 位有效地址寻址和 32 位有效地址寻址在基址、变址、比例因子上的区别。
21. 请说明指令中有效地址的表示形式。
22. 请说明直接寻址方式的定义和物理地址计算方法。
23. 请说明用符号地址表示直接寻址方式时应注意变量属性。
24. 请说明寄存器间接寻址方式的定义。
25. 请说明 16 位寄存器间接寻址中 BX、BP、SI、DI 的默认段寄存器。
26. 请说明寄存器相对寻址方式的定义和物理地址计算方法。
27. 请说明基址变址寻址方式的定义。
28. 请说明相对基址变址寻址方式的定义。
29. 请说明比例变址寻址方式对数组访问为什么有用。
30. 请说明与转移地址有关的寻址方式和与数据有关的寻址方式有什么区别。
31. 请说明无条件转移和条件转移指令在程序控制中的作用。
32. 请说明段内转移和段间转移的区别。
33. 请说明短转移、近转移和远转移的区别。

34. 请说明 CALL 指令和 JMP 指令的主要区别。
35. 请说明 MOV 指令的功能和常见限制。
36. 请说明 MOV 和 MOVS 指令的相同点和不同点。
37. 请说明 PUSH 和 POP 指令对堆栈的影响。
38. 请说明 XCHG 指令的功能。
39. 请说明 LEA 指令和 MOV 指令在取得地址方面的区别。
40. 请说明 CBW、CWD 等符号扩展指令的作用。
41. 请说明 ADD、ADC、SUB、SBB 指令的区别。
42. 请说明 CMP 指令为什么不保存减法结果但影响标志位。
43. 请说明 MUL 和 IMUL 指令的区别。
44. 请说明 DIV 和 IDIV 指令的区别。
45. 请说明 INC、DEC 指令和 ADD、SUB 指令对标志位影响的区别。
46. 请说明 AND、OR、XOR、NOT 指令的功能。
47. 请说明 TEST 指令和 AND 指令的区别。
48. 请说明 SHL、SHR、SAR 指令的区别。
49. 请说明 ROL、ROR、RCL、RCR 指令的区别。
50. 请说明字符串操作指令中 SI、DI、DS、ES 的默认约定。
51. 请说明 REP 前缀的作用。
52. 请说明 MOVS、LODS、STOS、CMPS、SCAS 指令的基本功能。
53. 请说明 DF 标志对字符串操作指令地址修改方向的影响。
54. 请说明 ARM64 中通用寄存器 Xn 和 Wn 的关系。
55. 请说明 ARM64 中 LDR 和 STR 指令的功能。
56. 请说明 ARM64 中 ADR 指令获得变量地址的原理。
57. 请说明 ARM64 中 NZCV 条件标志的含义。
58. 请说明 ARM64 中 MRS 和 MSR 指令的作用。
59. 请说明 ARM64 中有符号扩展和无符号扩展指令的区别。
60. 请说明 ARM64 中 ADD 和 ADDS、SUB 和 SUBS 的区别。
61. 请说明 ARM64 中 CMP 和 CMN 指令的功能。
62. 请说明 ARM64 中 B、BR、B.cond、CBZ、CBNZ、TBZ、TBNZ 的区别。
63. 请说明 ARM64 中 BL、BLR 和 RET 指令的作用。
64. 请说明 ARM64 中 SVC、HVC、SMC 和 ERET 指令的用途。

第四章 汇编语言程序格式

1. 请说明汇编程序 MASM/TASM 的主要功能。
2. 请说明连接程序 LINK/TLINK 的主要功能。
3. 请说明汇编源程序、目标程序和执行程序之间的生成关系。
4. 请说明汇编程序第一遍扫描的主要任务。
5. 请说明汇编程序第二遍扫描的主要任务。
6. 请说明汇编语言程序的指令可以分为哪三类。
7. 请说明机器指令、伪指令和宏指令的区别。
8. 请说明什么是伪操作。
9. 请说明处理器选择伪操作的作用。
10. 请说明缺省时汇编程序默认选择哪一种指令系统。
11. 请说明段定义伪操作 SEGMENT/ENDS 的格式和作用。
12. 请说明定义数据段、附加段、堆栈段和代码段时中间语句的区别。
13. 请说明段名是否说明段属性，为什么。

14. 请说明 SEGMENT 伪操作中定位类型 BYTE、WORD、DWORD、PARA、PAGE 的含义。
15. 请说明组合类型 PRIVATE、PUBLIC、COMMON、STACK 的区别。
16. 请说明连接程序如何根据段的组合类型合并多个模块中的段。
17. 请说明 USE16 和 USE32 的区别。
18. 请说明实模式下为什么应使用 USE16。
19. 请说明 ASSUME 伪指令的作用。
20. 请说明 ASSUME 是否会给段寄存器赋值。
21. 请说明使用段寄存器前为什么既要 ASSUME，又要用 MOV 等指令给段寄存器赋值。
22. 请说明数据定义伪操作 DB、DW、DD 的作用。
23. 请说明 DUP 操作符的作用。
24. 请说明变量名、标号、属性和偏移地址之间的关系。
25. 请说明 EQU 伪操作的作用。
26. 请说明 EQU 与数据定义伪操作的区别。
27. 请说明地址计数器的作用。
28. 请说明 ORG 伪操作对地址计数器的影响。
29. 请说明 EVEN 和 ALIGN 伪操作的作用。
30. 请说明基数控制伪操作的作用。
31. 请说明汇编语言源程序的一般语句格式。
32. 请说明名字项、操作项、操作数项和注释项的作用。
33. 请说明标号和变量的区别。
34. 请说明操作符 PTR、OFFSET、SEG、TYPE、LENGTH、SIZE 的作用。
35. 请说明汇编语言程序上机过程的一般步骤。
36. 请说明 EDIT、MASM、LINK、DEBUG 或 TD 在上机过程中的作用。
37. 请说明 EXE 文件和 COM 文件的主要区别。
38. 请说明 COM 文件为什么入口点必须是 100H。
39. 请说明 COM 程序装入内存时 CS、DS、ES、SS、IP、SP 的初始化特点。
40. 请说明 COM 文件适合编制较小程序的原因。
41. 请说明 COM 文件的上机步骤。
42. 请说明 ARM64 中 GNU as 和 GNU ld 的作用。
43. 请说明 ARM64 中 .section、.text、.data、.bss 的作用。
44. 请说明 ARM64 中 .type、.global、.size 的作用。
45. 请说明 ARM64 中数据定义伪操作 .byte、.hword、.word、.dword 的作用。
46. 请说明 ARM64 中 .equ 和 .equiv 的区别。
47. 请说明 ARM64 中位置计数器 “.” 的作用。
48. 请说明 ARM64 中 .align 和 .balign 的作用。
49. 请说明 ARM64 汇编语言程序的语句格式。
50. 请说明 ARM64 中标号和变量的定义方法。

第五章 循环与分支程序设计

1. 请简述汇编语言程序设计的基本步骤。
2. 请说明程序流程图中开始/结束框、功能执行框、判断框、方向线和子程序调用框的含义。
3. 请说明程序结构形式通常包括哪几种。
4. 请说明顺序程序结构的执行特点。
5. 请说明分支程序为什么要采用条件判断。
6. 请说明两分支结构和多分支结构的区别。
7. 请说明分支程序每次为什么只能执行其中一个分支。

8. 请说明循环程序设计适合解决哪类问题。
9. 请说明循环程序中循环体的含义。
10. 请说明循环程序的三个基本组成部分。
11. 请说明 DO-WHILE 结构和 DO-UNTIL 结构的区别。
12. 请说明循环程序设计中初始设置通常包括哪些内容。
13. 请说明循环体中通常需要完成哪些工作。
14. 请说明循环控制转移为什么要正确选择条件转移指令。
15. 请说明循环次数和 FLAGS 等其他条件在循环控制中的作用。
16. 请说明循环控制判断放在循环体前或循环体后的区别。
17. 请说明为什么循环程序不一定必须使用 LOOP 指令。
18. 请说明用计数值控制循环结束的基本方法。
19. 请说明在显示 BX 中二进制数的十六进制形式时，为什么每 4 位二进制数转换成 1 位十六进制字符。
20. 请说明把 0-9、A-F 转换成 ASCII 码的基本过程。
21. 请说明设计数据块设置程序时，为什么适合采用循环结构。
22. 请说明对连续内存单元操作时为什么常用变址寻址方式。
23. 请说明数据块移动程序中源地址指针、目的地址指针和计数器的作用。
24. 请说明已知循环次数和未知循环次数的循环控制方式有什么不同。
25. 请说明多重循环程序设计中内循环和外循环的关系。
26. 请说明多分支程序中跳转表的作用。
27. 请说明利用跳转表实现程序分支转移的基本思想。
28. 请说明逻辑尺中某一位为 1 时如何转移到相应分支。
29. 请说明实模式下发挥 80386 及其后继机型优势的基本方法。
30. 请说明 ARM 中实现分支程序常用哪些比较和跳转指令。
31. 请说明 ARM 中循环程序如何使用计数器和条件跳转实现。
32. 请说明 ARM 中用十六进制字符显示寄存器值的基本思路。
33. 请说明 ARM 汇编调用 printf 函数时 X0、X1 的参数传递约定。
34. 请说明 ARM Linux write 系统调用的主要参数。
35. 请说明返回 Linux 系统的函数返回方式和系统调用返回方式的区别。
36. 请说明统计二进制数中 “1” 的个数时，计数循环方式和优化查找方式的区别。

第六章 子程序结构

1. 请说明什么是子程序，又称为什么。
2. 请说明为什么需要子程序。
3. 请说明使用子程序的主要优点。
4. 请说明过程定义 PROC/ENDP 的格式和作用。
5. 请说明过程名和过程属性的含义。
6. 请说明 NEAR 属性和 FAR 属性的确定原则。
7. 请说明汇编程序如何根据过程属性确定 CALL 和 RET 指令属性。
8. 请说明调用程序和子程序在同一代码段中时应使用什么属性。
9. 请说明调用程序和子程序不在同一代码段中时应使用什么属性。
10. 请说明 FAR 属性子程序和 NEAR 属性子程序可被调用范围的区别。
11. 请说明子程序调用时 CALL 指令隐含使用堆栈完成哪些操作。
12. 请说明 near call 和 far call 保存返回地址的区别。
13. 请说明 RET 指令完成什么功能。
14. 请说明为什么子程序中要保存和恢复寄存器。
15. 请说明主程序和子程序使用寄存器冲突会产生什么问题。

16. 请说明保护和恢复寄存器的一般方法。
17. 请说明保存和恢复寄存器的次序为什么应该相反。
18. 请说明子程序中 PUSH 和 POP 指令为什么一般必须配对使用。
19. 请说明确定保护哪些寄存器的原则。
20. 请说明用寄存器向主程序回送结果时，该寄存器为什么不必保存。
21. 请说明 FLAGS 寄存器保存和恢复时应注意什么。
22. 请说明什么是参数传送。
23. 请说明调用程序和子程序之间参数传送的一般途径。
24. 请列举子程序参数传送的具体方法。
25. 请说明通过寄存器传送参数的特点和适用情况。
26. 请说明通过存储器直接访问传送参数的特点。
27. 请说明直接访问固定变量的子程序为什么通用性较差。
28. 请说明通过地址表传送变量地址的基本方法。
29. 请说明地址表传参为什么适用于参数较多的情况。
30. 请说明通过堆栈传送参数或参数地址的基本方法。
31. 请说明子程序嵌套调用和递归调用对堆栈有什么要求。
32. 请说明递归子程序为什么必须保存现场和返回地址。
33. 请说明 DOS 系统功能调用的一般格式。
34. 请说明 INT 21H 功能调用中 AH 功能号和入口/返回参数的作用。
35. 请说明 DOS 键盘输入、显示字符、显示字符串常见功能号。
36. 请说明设计子程序时应注意哪些问题。
37. 请说明初学者编写汇编程序的一般步骤。
38. 请说明 ARM64 函数/子程序的定义结构。
39. 请说明 ARM64 中 .global 声明对函数作用范围的影响。
40. 请说明 ARM64 中 bl、blr 和 ret 的作用。
41. 请说明 ARM64 函数调用时返回地址为什么保存在 X30。
42. 请说明 ARM64 函数嵌套调用时为什么需要保存 X30。
43. 请说明 ARM64 中自定义函数参数和返回值可以通过哪些方式传递。
44. 请说明 ARM64 中被 C 函数调用时参数传递规则。
45. 请说明 ARM64 中 X0-X7、X19-X28、X0 返回值的调用约定。
46. 请说明 ARM64 中使用堆栈传递参数时为什么要求 16 字节对齐。

第七章 高级汇编语言技术

1. 请说明什么是宏。
2. 请说明什么是宏指令。
3. 请说明宏指令和机器指令、伪指令在汇编语言程序中的关系。
4. 请说明子程序的优点和缺点。
5. 请说明宏汇编的用途。
6. 请说明在什么情况下使用宏汇编更加便利。
7. 请说明宏汇编相比子程序调用的优点和缺点。
8. 请说明宏定义、宏调用和宏展开的含义。
9. 请写出宏定义的一般格式。
10. 请说明 MACRO 和 ENDM 伪操作的作用。
11. 请说明哑元表、实元表和宏定义体的含义。
12. 请说明宏调用时实元和哑元如何对应。
13. 请说明实元数多于哑元数或少于哑元数时如何处理。

14. 请说明宏调用为什么必须先定义后调用。
15. 请说明宏展开时汇编程序完成哪些替换工作。
16. 请说明 LOCAL 中的标号在宏展开时如何处理。
17. 请说明哑元和实元为什么统称为变元。
18. 请说明宏定义是否可以无变元。
19. 请说明变元为什么可以是操作码或操作码的一部分。
20. 请说明宏汇编操作符 & 的作用。
21. 请说明宏注释 “;;” 与普通注释 “;” 的区别。
22. 请说明宏汇编操作符 % 的作用。
23. 请说明 :REQ 操作符的作用。
24. 请说明 := 操作符的作用。
25. 请说明宏指令名与指令助记符或伪操作名相同时会发生什么。
26. 请说明为什么建议宏指令名不要与指令助记符或伪操作名相同。
27. 请说明 LOCAL 伪操作解决什么问题。
28. 请说明 LOCAL 伪操作在宏定义体中的位置要求。
29. 请说明重复汇编的作用。
30. 请说明 REPT、IRP、IRPC 重复汇编伪操作的区别。
31. 请说明条件汇编的作用。
32. 请说明条件汇编为什么只在汇编时起作用。
33. 请说明高级语言结构在汇编语言中的作用。
34. 请说明 ARM64 中 .macro 和 .endm 的作用。
35. 请说明 ARM64 宏定义中参数引用为什么使用反斜杠。
36. 请说明 ARM64 宏展开列表中 “>” 和 “>>” 的含义。
37. 请说明 ARM64 宏定义可以无变元的情况。
38. 请说明 ARM64 宏定义中变元作为操作码一部分的实现方法。
39. 请说明 ARM64 宏定义中只能使用数字格式标号的原因。
40. 请说明 ARM64 中数字标号 Nb 和 Nf 的含义。
41. 请说明 ARM64 重复伪操作 .rept 的格式和作用。
42. 请说明 ARM64 不定重复伪操作 .irp 和 .irpc 的区别。
43. 请说明 ARM64 条件汇编 .if、.else、.endif 的作用。
44. 请说明条件汇编为什么不会生成机器指令。

第八章 输入输出程序设计

1. 请说明主机、外设和接口之间的关系。
2. 请说明从程序设计角度看，接口由哪些寄存器组成。
3. 请说明状态寄存器、控制寄存器和数据寄存器的作用。
4. 请说明什么是 I/O 端口地址。
5. 请说明 8086/8088 中 I/O 端口地址和存储单元地址的关系。
6. 请说明 8086/8088 CPU 提供的 I/O 端口地址空间大小。
7. 请说明 PC 及其兼容机实际使用的 I/O 端口地址范围。
8. 请说明存取接口寄存器中的数据依靠什么指令完成。
9. 请说明 IN 指令的格式和功能。
10. 请说明 OUT 指令的格式和功能。
11. 请说明 IN/OUT 指令为什么只能使用累加器。
12. 请说明 I/O 端口地址有哪两种给出方式。
13. 请列举 I/O 设备的数据传送方式。

14. 请说明选择 I/O 数据传送方式时主要考虑哪些因素。
15. 请说明程序直接控制 I/O 方式包括哪两种。
16. 请说明无条件传送方式的基本特点。
17. 请说明无条件传送方式适合什么情况。
18. 请说明查询方式的基本工作过程。
19. 请说明查询方式如何解决 CPU 与外设不同步的问题。
20. 请说明查询方式存在的主要问题。
21. 请说明查询等待中的“踏步”问题。
22. 请说明中断方式的基本工作过程。
23. 请说明中断方式的特点。
24. 请说明程序查询方式和程序中断方式在 CPU 与 I/O 并行性上的区别。
25. 请说明为什么需要 DMA 方式。
26. 请说明 DMA 方式的数据传输特点。
27. 请说明 DMA 方式主要用于哪些高速 I/O 设备。
28. 请说明 DMA 控制器在数据传送中的作用。
29. 请说明 DMA 数据传送控制过程的三个阶段。
30. 请说明 DMA 控制器的基本结构包括哪些寄存器。
31. 请说明声音输出程序中端口 61H 的第 1 位如何产生脉冲。
32. 请说明查询方式打印输出时为什么要先检查打印机状态。
33. 请说明中断的基本概念。
34. 请说明中断源可以分为哪些类型。
35. 请说明内部中断、外部中断和软件中断的区别。
36. 请说明中断向量和中断向量表的含义。
37. 请说明实模式下中断向量表的位置和每个中断向量占用的字节数。
38. 请说明中断类型号与中断向量表地址之间的关系。
39. 请说明中断响应时 CPU 自动完成哪些操作。
40. 请说明中断处理程序为什么通常用 IRET 返回。
41. 请说明 INT n 指令进入中断服务程序的过程。
42. 请说明软中断和子程序调用的区别。
43. 请说明用户自定义中断时需要完成哪些工作。
44. 请说明中断优先级的含义。
45. 请说明 80X86 中规定的中断优先级次序。
46. 请说明可屏蔽中断优先级默认由哪一芯片确定。
47. 请说明中断嵌套的含义。
48. 请说明使用 1CH 中断实现周期性工作的基本思想。
49. 请说明编写新的临时中断程序时为什么要保存和恢复原中断向量。
50. 请说明 80386 输入输出对 I/O 端口和 IN/OUT 指令的支持特点。
51. 请说明保护模式下 I/O 指令为什么带有特权级。
52. 请说明 CPL、DPL、RPL、IOPL 的含义。
53. 请说明执行 I/O 操作必须满足哪两个条件之一。
54. 请说明 80386 字符串 I/O 指令 INS 和 OUTS 的功能。
55. 请说明 80386 保护模式下中断处理和实模式下中断处理的区别。
56. 请说明中断门、陷阱门和任务门的作用区别。
57. 请说明保护模式下中断描述符表 IDT 和实模式下中断向量表的关系。

第九章 BIOS和DOS中断

1. 请说明 BIOS 的含义和存放位置。
2. 请说明 BIOS 中通常包含哪些内容。
3. 请说明 BIOS 为什么一般以中断处理程序形式存在并被调用。
4. 请列举常见 BIOS 软件中断功能调用的中断号。
5. 请说明 DOS 的含义。
6. 请说明 DOS 建立在 BIOS 基础上的含义。
7. 请说明 IBMBIO.COM 和 IBMDOS.COM 的主要作用。
8. 请列举常见 DOS INT 21H 功能调用。
9. 请说明应用程序、DOS、BIOS 和外设硬件接口之间的层次关系。
10. 请说明 DOS、BIOS 和直接访问硬件接口在功能强弱、硬件控制能力和编程复杂度上的区别。
11. 请说明编写应用程序时推荐的 I/O 调用顺序。
12. 请说明为什么通常 I/O 操作应首先选择 DOS 系统功能调用。
13. 请说明什么时候可以考虑 BIOS 调用。
14. 请说明应用程序直接操纵外设接口的优点和缺点。
15. 请说明 DOS 和 BIOS 功能调用采用什么方式实现。
16. 请说明应用程序进行 DOS/BIOS 中断调用时应明确哪些内容。
17. 请说明 DOS/BIOS 功能调用的一般步骤。
18. 请说明为什么 DOS 和 BIOS 功能调用使用中断方式实现调用。
19. 请从系统设计角度说明中断调用方式为什么简单、高效。
20. 请从用户编程角度说明中断调用方式为什么方便、可移植性好。
21. 请说明低层 BIOS 和 DOS 修改后为什么对上层应用透明。
22. 请说明键盘硬件由主板和键盘上的哪些部件配合完成。
23. 请说明键盘硬件中断 09H 的基本处理过程。
24. 请说明键盘软件中断和键盘硬件中断的区别。
25. 请说明字符码与扫描码的区别。
26. 请说明通码和断码的区别。
27. 请说明 BIOS 键盘中断处理程序如何处理扫描码。
28. 请说明键盘缓冲区的结构和特点。
29. 请说明 BIOS 键盘中断 INT 16H 的 0、1、2 号功能。
30. 请说明如何判断不具有 ASCII 码的功能键动作。
31. 请说明调用 DOS 或 BIOS 从键盘缓冲区读取字符后，缓冲区会发生什么变化。
32. 请说明 BIOS 调用和 DOS 调用读取键盘字符的区别。
33. 请说明清除键盘缓冲区后再读取一个字符的基本思路。
34. 请说明如何从键盘直接读取扫描码。
35. 请说明显示属性的含义。
36. 请说明显示缓冲区的作用。
37. 请说明为什么说显示屏幕是“存储器映像”。
38. 请说明直接写屏的含义。
39. 请说明文本方式下每个显示字符为什么占 2 个存储单元。
40. 请说明 25×80 单色显示文本方式下屏幕字符位置和显存容量的关系。
41. 请写出屏幕字符在显存中偏移地址的计算公式。
42. 请说明 BIOS 显示中断 INT 10H 可以完成哪些功能。
43. 请说明 INT 10H 实质上是对什么进行操作。
44. 请说明窗口上卷功能中 AL、BH、CH/CL、DH/DL 的作用。
45. 请说明利用 BIOS 显示中断清屏并设置光标位置的基本步骤。
46. 请说明 DOS 显示功能调用 INT 21H 中 AH=2、AH=6、AH=9 的区别。

47. 请说明 DOS 显示字符串为什么必须以 $ 结束。
48. 请说明显示字符串中如何实现自动换行。
49. 请说明打印机 I/O 可通过哪些 DOS 和 BIOS 中断完成。
50. 请说明主机输出给打印机的信息包括哪两类。
51. 请说明打印机常用控制字符如换页、回车、换行的作用。
52. 请说明 DOS 打印一个字符的调用方法。
53. 请说明 BIOS 打印功能 INT 17H 如何选择打印机。
54. 请说明将键盘接收字符同时显示并打印的程序基本流程。
55. 请说明系统定时器 08H 中断和用户 1CH 中断之间的关系。
56. 请说明 BIOS 为什么在 08H 中断处理程序中安排 INT 1CH。
57. 请说明利用 1CH 中断显示实时时钟的基本思想。
58. 请说明主程序在使用新的 1CH 中断处理程序前后需要完成哪些工作。
59. 请说明中断处理程序中为什么要保护现场并用 IRET 返回。
60. 请说明将 BCD 时间数据转换成 ASCII 码的基本方法。

参考答案

第二章参考答案

1. 计算机系统主要由存储器、运算器、控制器、输入设备和输出设备构成。
2. 分散连接是各部件之间使用单独连线；总线连接是各部件连接到一组公共信息传输线，便于 I/O 设备与主机灵活连接。
3. 数据总线传输数据；地址总线传输存储单元或 I/O 设备地址；控制总线控制 CPU 与其他部件的通信方式。
4. 内部数据总线宽度是 CPU 芯片内部数据传送宽度；外部数据总线宽度是 CPU 与外部交换数据时的数据宽度。
5. 80X86 是 Intel 开发的微处理器系列，经历 8086、80286、80386、80486、Pentium 等，性能和功能逐步增强。
6. 高级语言经编译可适配不同 CPU；汇编语言与机器指令一一对应，对不同 CPU 不兼容，但程序设计方法通用。
7. 字长增加有利于提高计算精度和速度。
8. 地址总线宽度决定可表示的地址数，如 20 根地址线可寻址 2^20=1MB。
9. 80286 增加保护模式，可提供虚拟存储管理和多任务管理。
10. 虚拟 8086 模式可在一台机器上同时模拟多个 8086 处理器工作，并支持安全隔离。
11. 把协处理器集成到 CPU 中可提高浮点处理速度。
12. 8086 只有实模式；80286 有实模式和保护模式；80386 起有实模式、保护模式和虚拟 8086 模式。
13. CPU 复位或加电后首先进入实模式。
14. 实模式下由 16 位段寄存器和 16 位偏移地址形成 20 位物理地址。
15. 实模式没有特权级和保护机制，所有段都可读、写和执行。
16. 保护模式提供多任务、内存分段分页管理、特权级保护和虚拟内存等功能。
17. 实模式下应用程序可执行任何 CPU 指令、读写所有内存，DOS 无法限制其行为。
18. 保护模式通过特权级和分段分页，使应用程序只能访问属于自己的内存空间。
19. 保护模式下段大小可达 4GB，支持分页、虚拟内存、多任务和保护机制。
20. 虚拟 8086 模式以任务形式在保护模式下执行，支持任务切换和内存分页。
21. EL0 通常为用户程序，EL1 为操作系统内核，EL2 为虚拟机监视器，EL3 为低级固件。

22. Non-secure state 称为 Normal World，secure state 称为 Secure World。
23. AArch32 是 32 位执行状态，兼容 Armv7-A；AArch64 是 64 位执行状态，使用 A64 指令集。
24. 包括 Aborts、Reset、异常生成指令 SVC/HVC/SMC，以及 IRQ、FIQ、SError 等中断。
25. 由存储器、微处理器、I/O 接口电路、输入/输出设备以及地址/数据/控制总线构成。
26. 数据寄存器用于数据传送，状态寄存器用于查看外设状态，命令寄存器用于控制外设工作。
27. 单独编址用专用 I/O 指令访问；统一编址用访存指令访问。
28. 通用寄存器用于数据和运算；段寄存器保存段地址；指针和变址寄存器常用于堆栈、数组和字符串寻址。
29. 状态标志反映运算结果状态；控制标志控制 CPU 某些操作方式。
30. CF 表示进位/借位，ZF 表示结果为零，SF 表示符号位，OF 表示有符号溢出。
31. 分段便于程序代码、数据、堆栈分开管理，并支持较灵活的逻辑地址表示。
32. 逻辑地址为段地址:偏移地址，段地址左移 4 位形成段基址，再加偏移地址得物理地址。
33. PA = 段寄存器内容 × 10H + 偏移地址。
34. CS 指代码段，DS 指数据段，SS 指堆栈段，ES 指附加段。
35. 指令取 CS:IP，堆栈默认 SS:SP，普通变量默认 DS:EA，源串默认 DS:SI，目标串默认 ES:DI，BP 默认 SS:EA。
36. 汇编指令给出逻辑地址，机器自动按当前模式进行地址转换得到物理地址。
37. 各段基地址和长度可独立设置，因此段之间可以相邻、相离或重叠。
38. 为了解决大地址空间寻址、多任务隔离和虚拟存储器支持问题。
39. 保护模式逻辑地址由选择器和偏移地址组成。
40. 选择器只用于查描述符表，段基地址存放在描述符中，不能由选择器直接表示。
41. 用选择器查描述符表得到段描述符和段基地址，再加偏移地址得到物理地址。
42. 实模式用段寄存器左移 4 位加偏移；保护模式用选择器查描述符获得段基址再加偏移。
43. 描述符长 8 字节，由段基地址、界限、访问权限和附加字段组成，用来说明段的位置、大小、控制和状态信息。
44. 段基地址说明段位置，界限说明段大小，访问权限说明可访问方式和特权，附加字段说明粒度、默认操作数大小等。
45. INDEX 指描述符表中描述符索引，TI 指 GDT/LDT，RPL 指请求访问特权级。
46. GDT 是全局描述符表，LDT 是局部描述符表，IDT 是中断描述符表。
47. 程序不可见寄存器只能由操作系统或硬件管理，如 GDTR、LDTR、IDTR。
48. 描述符高速缓冲存储器缓存正在使用段的描述符，减少多次访问主存，提高寻址效率。
49. 大端高位在低地址，小端低位在低地址；X86 采用小端模式。
50. MMU 负责把虚拟地址映射为物理地址，并提供内存访问授权和多任务支持。

第三章参考答案

1. 指令系统是计算机所能执行的全部机器指令的集合。
2. CPU 依靠机器指令计算和控制系统，指令系统规定了 CPU 能完成的全部基本操作。
3. 零地址指令无显式操作数，一地址指令有一个操作数，二地址指令通常含目的和源操作数。
4. 熟记操作码助记符，正确给出操作数。
5. 寻址方式是指令指定操作数地址的方式。
6. 操作数可在指令中、CPU 寄存器中、内存单元中或 I/O 接口寄存器中。
7. OP 为操作码，dst 为目的操作数，src 为源操作数。
8. 立即寻址操作数在指令中给出，常用于给寄存器赋初值，只能作源操作数且字长要一致。
9. 寄存器寻址的操作数在指定寄存器中。
10. 立即数是常量，只能被读取作为源，不能作为可写目的位置。
11. 先由寻址方式求出内存有效地址，再结合段基地址得到物理地址读取操作数。

12. 实模式段基址为段寄存器内容乘 16；保护模式段基址由段选择子查描述符得到。
13. 段寄存器名:偏移地址。
14. 缺省时使用约定段寄存器；特别指定用于跨段访问，但要遵守规则。
15. 指令 CS:IP，堆栈 SS:SP，普通变量 DS:EA，源串 DS:SI，目标串 ES:DI，BP 作基址默认 SS:EA。
16. 有效地址 EA 是操作数在段内的偏移地址。
17. 基址、变址、比例因子、位移量。
18. 基址常指数组/字符串首地址，变址指定元素或字符，比例因子适合元素大小换算，位移量是指令中给出的地址偏移。
19. EA = 基址 + 变址 × 比例因子 + 位移量。
20. 16 位寻址基址为 BX/BP，变址为 SI/DI，无比例因子；32 位寻址基址可用通用寄存器，变址除 ESP 外，比例因子为 1/2/4/8。
21. [基址+变址*比例因子+位移量]，完整地址可写为段寄存器名:[...]。
22. 直接寻址中 EA 由指令直接给出，默认 PA=DS×16+EA，也可用段超越。
23. 要注意变量类型属性与指令操作数字长一致，必要时用 PTR 指明。
24. 寄存器间接寻址中 EA 存在 BX、BP、SI、DI 或 32 位寄存器中。
25. BX、SI、DI 默认 DS；BP 默认 SS。
26. EA 为基址/变址寄存器内容加位移量，BX/SI/DI 默认 DS，BP 默认 SS。
27. EA 由一个基址寄存器和一个变址寄存器内容相加得到。
28. EA 由基址、变址和位移量共同组成。
29. 通过变址寄存器乘 1/2/4/8，可方便访问 1、2、4、8 字节数组元素。
30. 数据寻址确定操作数位置；转移地址寻址确定下一条执行指令地址。
31. 无条件转移直接改变执行顺序；条件转移根据标志或条件改变执行顺序。
32. 段内转移只改变 IP/EIP；段间转移同时改变 CS 和 IP/EIP。
33. 短转移位移范围较小，近转移在同一代码段内，远转移可跨代码段。
34. JMP 只转移；CALL 转子程序前还保存返回地址。
35. MOV 将源操作数送目的操作数，通常不能内存到内存，CS 不能用 MOV 直接改变。
36. MOV 是一般数据传送；MOVS 是字符串传送，按 DS:SI 到 ES:DI 传送并自动修改 SI/DI。
37. PUSH 先修改 SP 再入栈；POP 从栈顶取出再修改 SP。
38. XCHG 交换两个操作数内容。
39. LEA 取有效地址送寄存器；MOV 取操作数内容。
40. CBW/CWD 等把较短有符号数扩展成较长有符号数。
41. ADD 加法，ADC 带进位加；SUB 减法，SBB 带借位减。
42. CMP 用减法设置标志供判断，但不保存结果。
43. MUL 为无符号乘法，IMUL 为有符号乘法。
44. DIV 为无符号除法，IDIV 为有符号除法。
45. INC/DEC 加减 1，但不影响 CF；ADD/SUB 会影响 CF。
46. AND 按位与，OR 按位或，XOR 按位异或，NOT 按位取反。
47. TEST 只按位与并设置标志，不保存结果；AND 保存结果。
48. SHL 逻辑左移，SHR 逻辑右移补 0，SAR 算术右移保持符号位。
49. ROL/ROR 循环移位不含 CF；RCL/RCR 通过 CF 循环移位。
50. 源串默认 DS:SI，目标串默认 ES:DI。
51. REP 按 CX/ECX 重复执行字符串指令。
52. MOVS 传送串，LODS 取串元素到累加器，STOS 存累加器到串，CMPS 比较串，SCAS 扫描串。
53. DF=0 地址递增，DF=1 地址递减。
54. Xn 是 64 位寄存器，Wn 是其低 32 位。
55. LDR 从内存装入寄存器，STR 将寄存器内容存入内存。

56. ADR 将 PC 附近标签地址通过 PC 相对位移计算后送寄存器。
57. N 为负，Z 为零，C 为进位/无借位，V 为溢出。
58. MRS 读特殊寄存器到通用寄存器，MSR 写通用寄存器到特殊寄存器。
59. 有符号扩展按符号位扩展，无符号扩展高位补 0。
60. ADD/SUB 不更新 NZCV；ADDS/SUBS 完成运算后更新 NZCV。
61. CMP 做减法不保存结果并更新 NZCV；CMN 做加法意义的比较并更新 NZCV。
62. B 直接转移，BR 寄存器间接转移，B.cond 条件转移，CBZ/CBNZ 比较寄存器是否为 0 后转移，TBZ/TBNZ 测试指定位后转移。
63. BL 直接调用并把返回地址存 X30；BLR 间接调用并存 X30；RET 用 X30 或指定寄存器返回。
64. SVC/HVC/SMC 分别调用 EL1/EL2/EL3 服务；ERET 从异常返回。

第四章参考答案

1. 检查程序、报告语法错误、展开宏指令、生成目标程序和列表文件。
2. 连接目标模块、段合并、解决外部调用、产生可重定位装入模块和可选内存映像文件。
3. ASM 经汇编生成 OBJ，OBJ 经连接生成 EXE。
4. 建立符号表、处理伪操作、估算指令长度并维护地址计数器。
5. 求操作数表达式值、检查段跨越、生成机器指令和初值。
6. 机器指令、伪指令、宏指令。
7. 机器指令生成机器代码运行时执行；伪指令仅汇编时处理；宏指令汇编时展开为一组指令。
8. 伪操作是告诉汇编程序的功能说明或定义，不生成机器指令。
9. 告诉汇编程序选择哪一种处理器指令系统。
10. 默认选择 8086 指令系统。
11. 用 segment_name segment ... segment_name ends 定义一个段。
12. 数据/附加/堆栈段中只能包括伪指令；代码段中可有机器指令和相关伪指令/宏指令。
13. 段名只是便于阅读的名字，不说明段属性。
14. BYTE 任意字节，WORD 偶地址，DWORD 4 的倍数，PARA 小段边界，PAGE 256 字节页边界。
15. PRIVATE 不合并；PUBLIC 同名段连接；COMMON 同名段重叠；STACK 同名栈段组成一个堆栈段。
16. LINK 根据 SEGMENT 的组合类型将多个模块同名段连接、重叠或组成栈段。
17. USE16 为 16 位偏移寻址段长 64KB；USE32 为 32 位偏移寻址段长 4GB。
18. 实模式使用 16 位偏移地址，应使用 USE16。
19. ASSUME 明确段和段寄存器之间的对应关系。
20. ASSUME 不给段寄存器赋值。
21. ASSUME 只给汇编程序提供约定，运行时还必须用 MOV 等指令实际装入段寄存器。
22. DB 定义字节，DW 定义字，DD 定义双字并分配存储单元。
23. DUP 用于重复分配或初始化多个相同数据项。
24. 变量名/标号是符号地址，属性说明类型，偏移地址由地址计数器确定。
25. EQU 给表达式命名，相当于定义常量。
26. EQU 不分配存储单元；数据定义会分配存储单元并可初始化。
27. 地址计数器记录当前段中下一条指令或数据的偏移地址。
28. ORG 改变地址计数器，使后续内容从指定偏移开始。
29. EVEN 使下一个地址为偶地址；ALIGN 使地址按指定边界对齐。
30. 控制数值常量默认进制。
31. [名字项] 操作项 [操作数项] [;注释]。
32. 名字项定义标号/变量，操作项说明指令或伪操作，操作数项给出参数，注释解释程序。
33. 标号表示指令地址；变量表示数据项地址并有类型属性。

34. PTR 指明类型，OFFSET 取偏移，SEG 取段地址，TYPE 取类型长度，LENGTH 取元素个数，SIZE 取总字节数。
35. 编辑源程序、汇编、连接、执行、调试。
36. EDIT 编辑，MASM 汇编，LINK 连接，DEBUG/TD 调试。
37. EXE 可多段、可重定位；COM 只有一个段、入口 100H、总空间不超 64KB。
38. COM 程序装入在 PSP 后，PSP 长 100H 字节，代码从偏移 100H 开始。
39. CS、DS、ES、SS 指向 PSP 段基址，IP=100H，SP 指向段末栈顶。
40. COM 占内存少、速度快，适合小程序。
41. EDIT、MASM、LINK/T、EXE2BIN、执行 COM、DEBUG/TD 调试。
42. GNU as 汇编源程序生成目标文件，GNU ld 连接生成可执行程序。
43. .section 定义段；.text 代码段；.data 初始化数据段；.bss 未初始化数据段。
44. .type 定义符号类型，.global 声明全局符号，.size 给出函数/对象长度。
45. 分别定义 1、2、4、8 字节数据。
46. .equ 定义常量；.equiv 在名称已定义时会报错。
47. “.” 表示当前位置计数器。
48. .align 按低位零个数对齐；.balign 按指定字节边界对齐。
49. [标号:] 操作项 [操作数] [//注释]。
50. 标号以冒号结束；变量在数据段用标号加 .byte/.short/.word 等定义。

第五章参考答案

1. 分析问题、确定算法、绘制流程图、分配存储空间、编写源程序、上机调试。
2. 开始/结束表示程序起止；功能框表示执行操作；判断框表示条件分支；方向线表示流程；子程序框表示调用。
3. 顺序、循环、分支、子程序及复合结构。
4. 从第一条可执行指令开始，按编写顺序逐条执行到最后。
5. 分支程序根据判断条件转向不同处理。
6. 两分支通常用条件转移；多分支常用无条件转移、变址、间接寻址或跳转表。
7. 因为一次判断只能选择满足条件的某一路处理。
8. 适合处理算法相同、数据或地址有规律变化的问题。
9. 被重复执行多次的指令段称为循环体。
10. 初始设置、循环体、循环控制转移。
11. DO-WHILE 先判断满足再执行；DO-UNTIL 先执行再判断直到条件满足。
12. 循环次数、数据初值、地址/变址指针初始化等。
13. 完成算法处理、修改数据或地址指针、设置循环控制条件。
14. 条件转移决定循环是否继续，选错会导致次数错误或死循环。
15. 循环次数控制重复次数；FLAGS 等条件可控制是否提前结束。
16. 放前可避免一次也不该执行的循环；放后至少执行一次，性能也可能不同。
17. 可用 DEC/JNZ、CMP/Jcc 等计数控制，不一定必须用 LOOP。
18. 设置计数器，每循环一次加/减，判断是否达到结束条件。
19. 一位十六进制数正好对应 4 位二进制数。
20. 取低 4 位，加 30H 得 0-9；若大于 39H 再加 7 得 A-F。
21. 连续单元设置重复同类操作，数据和地址均规律变化，适合循环。
22. 变址寄存器可作为地址指针，逐次修改指向连续内存单元。
23. 源指针指向被传送数据，目的指针指向目标单元，计数器控制传送次数。
24. 已知次数用计数器控制；未知次数用查找结果或条件标志控制。
25. 外循环每执行一次，内循环完整执行一轮，需分别设置和保存计数。

26. 跳转表保存各分支入口地址，用索引选择并转移到对应分支。
27. 根据选择标志计算表项地址，取出分支入口地址后间接转移。
28. 逐位测试逻辑尺，为 1 时用对应表项地址转移到对应程序。
29. 在实模式下使用 80386 新增寄存器和指令，但仍按实模式地址规则编程。
30. cmp、bgt、blt、beq、bne 等。
31. 设置计数器，循环体后用 subs 更新计数并用条件转移继续或结束。
32. 每次取 4 位转换成 ASCII，调用 printf 或 write 显示，循环 16 次。
33. X0 保存格式字符串地址，X1 保存第一个待显示数据。
34. X8 为系统调用号，X0 为输出文件，X1 为字符串地址，X2 为长度，svc #0 调用。
35. 函数返回定义 main、保存/恢复 X29/X30、ret；系统调用返回用 exit 系统调用号和 svc #0。
36. 计数循环固定检查每一位；优化查找用 clz/cls 跳过连续 0/1 位串，减少循环次数。

第六章参考答案

1. 子程序又称过程，相当于高级语言中的过程或函数。
2. 为了程序段共享、模块化设计、简化程序设计、节省存储空间。
3. 节省存储空间，减少程序设计时间。
4. procedure_name PROC Attribute ... procedure_name ENDP，用于定义过程。
5. 过程名是子程序入口符号地址；属性说明 NEAR 或 FAR。
6. 同一代码段用 NEAR；不同代码段用 FAR。
7. 汇编程序根据过程属性确定 CALL/RET 是近调用/返回还是远调用/返回。
8. NEAR 属性。
9. FAR 属性。
10. FAR 可被同段或不同段调用；NEAR 只能被同一代码段调用。
11. 保存返回地址并转入子程序。
12. near call 保存 IP；far call 保存 CS 和 IP。
13. RET 从堆栈恢复返回地址并返回调用程序。
14. 子程序是独立共享模块，会与主程序发生寄存器使用冲突。
15. 子程序改变主程序正在使用的寄存器会导致主程序结果错误。
16. 进入子程序用 PUSH 保存，返回前用 POP 恢复。
17. 堆栈后进先出，所以恢复顺序必须与保存顺序相反。
18. 保持堆栈平衡，避免返回地址或参数错乱。
19. 保存子程序将使用且调用程序仍需要的寄存器和标志寄存器。
20. 回送结果的寄存器必须保留新结果，保存后恢复会破坏返回值。
21. FLAGS 保存优先，恢复时通常最后恢复。
22. 参数传送是调用程序和子程序之间的信息传送。
23. 通过寄存器或存储器。
24. 寄存器、存储器、地址表、堆栈。
25. 使用方便，适合参数较少的情况。
26. 子程序直接访问数据段变量，简单但变量固定。
27. 处理单元在子程序中固定约定，换数组时不能复用同一子程序。
28. 先建立由参数地址组成的地址表，再把表首地址传给子程序。
29. 多个参数只需传表首地址，子程序可按表项取得各参数地址。
30. 主程序压入参数或地址，子程序按栈位置取用，返回后主程序调整 SP。
31. 必须正确保存返回地址、局部状态和参数，保持堆栈平衡。
32. 递归多次调用同一过程，每层都要有独立现场和返回地址。
33. 设置入口参数，AH=功能号，INT 21H，取得返回参数。

34. AH 指明功能；入口参数按规定送寄存器；返回参数由 DOS 放入规定寄存器。
35. AH=1 输入并回显，AH=2 显示字符，AH=9 显示字符串，AH=0AH 输入字符串。
36. 功能定义说明、参数传递、寄存器保存恢复、堆栈状态。
37. 理解题意、画流程图、查指令和调用规定、写结构、优化、静态模拟查错。
38. .global、.type、函数标签、函数体、.size。
39. .global 声明后可被其他文件调用；未声明通常只在本文件内可见。
40. bl 直接调用，blr 间接调用，ret 返回。
41. ARM64 调用指令把下一条指令地址存入 X30 作为返回地址。
42. 嵌套调用会覆盖 X30，不保存则无法返回原调用者。
43. 可通过寄存器、内存单元或堆栈传递。
44. 前 8 个参数用 X0-X7，超过的参数放堆栈，返回值用 X0。
45. X0-X7 传参，X19-X28 由被调用函数保存，X0 传返回值。
46. ARM64 调用约定要求 SP 16 字节对齐，通常一次压入两个 64 位寄存器。

第七章参考答案

1. 宏是源程序中一段有独立功能的程序代码。
2. 宏指令是用户自定义的指令，用一条宏调用代替一段程序代码。
3. 汇编语言程序可由机器指令、伪指令和宏指令组成。
4. 子程序优点是模块化、省内存、便于调试修改；缺点是调用返回、寄存器保护和参数传送有开销。
5. 宏汇编使短小或参数较多的功能使用更方便、更灵活、更接近高级语言。
6. 子程序本身较短或需要传送参数较多时。
7. 宏参数传送简单、执行效率高；缺点是多次展开会占用更多内存。
8. 宏定义写出宏体；宏调用使用宏名；宏展开是汇编时用宏体替换宏调用。
9. macro_name MACRO [哑元表] [LOCAL 标号表] ... ENDM。
10. MACRO 开始宏定义，ENDM 结束宏定义。
11. 哑元表是形参，实元表是实参，宏定义体是一组有独立功能的语句。
12. 实元与哑元按位置一一对应。
13. 实元多余则多余无效；实元不足则多余哑元为空。
14. 汇编器必须先知道宏定义，才能展开宏调用。
15. 复制宏定义体到调用处，并用实元替换哑元。
16. LOCAL 标号被替换成唯一的 ??xxxx 标号。
17. 它们都是宏替换中参与传递的参数。
18. 可以，宏体不需要参数时可无变元。
19. 宏展开本质是文本替换，实元可以替代操作码或其一部分。
20. & 合并前后符号形成一个符号。
21. ";" 注释会随宏展开出现；";;" 注释不展开。
22. % 使表达式立即求值，并用结果替换哑元。
23. :REQ 指定某个变元必须提供实元。
24. := 为宏变元提供缺省值。
25. 宏定义优先级最高，同名助记符或伪操作会被认为是宏指令。
26. 避免覆盖原指令或伪操作含义导致程序混乱。
27. 解决宏体标号多次展开时重复定义的问题。
28. 必须是 MACRO 后第一个语句，中间不能有注释或分号标志。
29. 汇编时重复生成一段语句，减少源程序重复书写。
30. REPT 按次数重复；IRP 对实元序列逐项重复；IRPC 对字符串逐字符重复。
31. 根据汇编时条件决定某段源程序是否参加汇编。

32. 条件判断由汇编器执行，不是运行时指令。
33. 用类似高级语言的结构组织汇编程序，提高可读性和编程效率。
34. .macro 开始 ARM64 宏定义，.endm 结束宏定义。
35. GNU as 用 \参数名 表示宏参数替换。
36. “>” 表示宏展开一层，“>>” 表示嵌套展开两层。
37. 如保存/恢复固定寄存器的 enter_func、exit_func 宏。
38. 通过 \参数 与操作码文本拼接，如 b\cc。
39. 普通标号多次展开会重复定义，数字标号可按最近前后引用避免冲突。
40. Nb 引用前面最近的 N:，Nf 引用后面最近的 N:。
41. .rept count ... .endr 按 count 次重复汇编块。
42. .irp 对参数列表中每个元素重复；.irpc 对字符串每个字符重复。
43. .if 判断条件，.else 给出假分支，.endif 结束条件汇编。
44. 条件汇编只控制源语句是否被汇编，本身不是机器指令。

第八章参考答案

1. 外设通过硬件接口或控制器与 CPU 相连。
2. 接口由状态寄存器、控制寄存器和数据寄存器组成。
3. 状态寄存器反映外设状态，控制寄存器控制外设工作，数据寄存器传送数据。
4. 系统分配给 I/O 接口寄存器的访问地址称为 I/O 端口地址。
5. 二者各自独立，占用不同地址空间。
6. 64KB，可接 64K 个 8 位端口或 32K 个 16 位端口。
7. 0-3FFH。
8. IN 和 OUT 指令。
9. IN 将端口数据输入到 AL/AX，端口可为立即数或 DX。
10. OUT 将 AL/AX 输出到端口，端口可为立即数或 DX。
11. 80X86 I/O 指令规定端口数据只能经累加器传送。
12. 8 位立即数端口 PORT 或 DX 间接端口地址。
13. 程序直接控制 I/O、中断方式、DMA 方式。
14. CPU 与 I/O 速度匹配、减轻 CPU 负担、外设请求服务处理时机。
15. 无条件传送方式和查询方式。
16. 默认外设总是准备好或不忙，直接用 IN/OUT 传送。
17. 外设速度与 CPU 同步且 CPU 负载轻时。
18. 先读状态端口，若准备好/不忙再传送，否则循环等待。
19. 通过反复查询状态，等外设准备好后再传送。
20. 有踏步等待，CPU 与 I/O 串行工作，效率低。
21. CPU 反复执行查询等待，不能做其他有效工作。
22. 外设准备好后主动请求中断，CPU 暂停当前程序，执行服务程序后返回。
23. CPU 与 I/O 可并行，能及时响应意外事件或异常。
24. 查询方式串行并有等待；中断方式并行且无踏步等待。
25. 大批量高速传输时减少 CPU 开销，避免高速设备丢失数据。
26. 以数据块为单位，CPU 和外设并行，传送过程基本不需 CPU 干预。
27. 网卡、磁带、磁盘、模数转换器等。
28. 生成访存地址并控制访存过程，使 I/O 与内存直接成批传送。
29. CPU 初始化 DMA；DMA 控制设备与内存块交换；结束后 CPU 后处理。
30. 控制寄存器、状态寄存器、地址寄存器、字节计数器。
31. 使端口 61H 第 1 位交替为 0 和 1 产生脉冲，延迟控制频率。

32. 只有打印机不忙、可接收时才能输出，否则数据可能丢失或出错。
33. 中断是 CPU 暂停当前程序，转去处理事件，处理完再返回的机制。
34. 可分内部中断、外部中断和软件中断。
35. 内部中断由 CPU 内部异常产生；外部中断由外设请求产生；软件中断由 INT 指令产生。
36. 中断向量是服务程序入口地址；中断向量表保存各类中断向量。
37. 位于 00000H 开始，每个中断向量占 4 字节。
38. 中断类型号 n 的向量地址为 n×4。
39. 保存 FLAGS、CS、IP，清除 IF/TF，取中断向量转入服务程序。
40. IRET 恢复 IP、CS、FLAGS，回到被中断程序。
41. INT n 按 n×4 查向量表，取得入口 CS:IP 并转入服务程序。
42. 软中断通过中断向量转入且保存 FLAGS；子程序调用由 CALL 预先安排并保存返回地址。
43. 编写中断服务程序，设置中断向量，正确设置中断允许、屏蔽位和优先级。
44. 多个中断同时请求时 CPU 确定服务先后次序。
45. 软件中断、非屏蔽中断、可屏蔽中断、单步中断。
46. 8259A 中断控制器。
47. 正在执行中断服务程序时又被其他中断请求打断。
48. 系统定时器每秒约 18.2 次调用 INT 1CH，用户改写 1CH 向量实现周期任务。
49. 避免破坏系统或其他程序原有 1CH 功能，程序结束时恢复环境。
50. 支持 65536 个 I/O 端口，IN/OUT 仍只能用 AL/AX/EAX。
51. 为保护系统，低特权级程序不能随意执行 I/O 指令。
52. CPL 当前特权级，DPL 描述符特权级，RPL 请求特权级，IOPL 为 I/O 操作特权级。
53. CPL≤IOPL，或 TSS 的 I/O 允许位图允许访问该端口。
54. INS 从 DX 端口输入到 ES:EDI；OUTS 从内存串输出到 DX 端口，可配合 REP。
55. 实模式使用中断向量表；保护模式使用 IDT 中的门描述符并进行保护检查。
56. 中断门和陷阱门给出处理程序入口，任务门引发任务切换。
57. IDT 相当于保护模式下的中断向量表，通过门描述符获得中断处理入口。

第九章参考答案

1. BIOS 是固化在 ROM 中的基本输入输出系统，地址空间示例为 FE000H-FFFFFH。
2. 包含主要 I/O 设备处理和接口控制程序、硬件/软件中断处理程序及常用系统例行程序。
3. BIOS 功能多与硬件中断和底层 I/O 相关，用中断形式便于统一调用。
4. INT 10H 显示，13H 磁盘，14H 串口，16H 键盘，17H 打印，19H 引导，33H 鼠标。
5. DOS 是磁盘操作系统。
6. DOS 通过 BIOS 控制硬件，是建立在 BIOS 之上的 PC 操作系统。
7. IBMBIO.COM 负责 I/O 设备处理；IBMDOS.COM 负责作业管理、文件管理、设备管理等。
8. AH=1 键盘输入回显，2 显示字符，9 显示字符串，0AH 缓冲输入，4CH 终止。
9. 应用程序在上层，下面依次是 DOS、BIOS、外设硬件接口和外设。
10. 层次越低硬件控制能力越强、效率可能越高，但编程难度和复杂度越大。
11. DOS -> BIOS -> 直接外设接口访问。
12. DOS 调用容易、硬件依赖少、可移植性好。
13. DOS 不提供某种服务或不能使用 DOS 时。
14. 直接访问接口效率最高、控制最强，但编程复杂、硬件相关性强。
15. 采用软件中断方式。
16. 中断类型号、功能号/子功能号、入口参数和返回参数。
17. 装入入口参数，设置 AH/AL 功能号，执行 INT n，检查返回参数。
18. 因为中断方式与硬件中断处理入口统一，系统设计方便，对上层透明。

19. 不必同时设置中断入口和子程序入口，功能调用与硬件中断处理机制统一。
20. 应用程序不必关心低层入口地址和现场保存细节，程序独立、可移植。
21. 上层只通过固定中断号和功能号调用，低层入口地址变化由中断向量处理。
22. 主板上有 8255 接口和 8259 中断控制；键盘由 8048 单片机扫描矩阵。
23. 按键且中断允许时发 09H 中断，BIOS 转换扫描码为字符码并放入键盘缓冲区。
24. 硬件中断处理按键扫描和缓冲；软件中断供应用程序从缓冲区读取键盘信息。
25. 扫描码表示物理按键位置；字符码表示转换后的 ASCII 或功能键码。
26. 通码表示按下键，断码表示放开键。
27. BIOS 将扫描码转换为 ASCII 字符码或 0，并与扫描码一起存入键盘缓冲区。
28. 键盘缓冲区是先进先出的循环队列，保存字符码和扫描码。
29. AH=0 读字符；AH=1 检测缓冲区；AH=2 取键盘状态字。
30. 用 INT 16H AH=2 读键盘状态字判断变换键状态。
31. 读出后相应键盘字符会从缓冲区清除。
32. BIOS INT 16H 返回字符码和扫描码；DOS INT 21H 功能更方便，通常返回字符码。
33. 循环检测缓冲区，非空则读走字符，直到为空，再等待新输入。
34. 可关闭键盘中断用查询读 60H，或改写 09H 中断向量由自定义程序读取扫描码。
35. 显示属性定义字符的闪烁、反向、高亮度、前景色和背景色等。
36. 显示缓冲区存放屏幕文本代码、属性或图形信息。
37. 显存作为系统存储器一部分，可像普通内存一样访问，内容映射到屏幕。
38. 直接写屏是直接存取显示存储器内容实现显示。
39. 一个字节存字符码，另一个字节存显示属性。
40. 25×80=2000 个字符，每字符 2 字节，需约 4000B；16KB 可存 4 页。
41. Char_offset=page_offset+((row*width)+column)*byte。
42. 设置显示模式、光标类型/位置、读光标位置、读写字符属性、选择页、滚屏等。
43. 实质是对显示存储区的操作。
44. AL 为上卷行数，BH 为空白属性，CH/CL 为左上角行列，DH/DL 为右下角行列。
45. 用 INT 10H AH=6 AL=0 清窗口，再用 AH=2 设置光标到左上角。
46. AH=2 显示字符并检查 Ctrl-Break；AH=6 显示字符不检查；AH=9 显示 $ 结尾字符串。
47. DOS AH=9 以 $ 作为字符串结束标志。
48. 在 $ 前加入回车 0DH 和换行 0AH。
49. DOS INT 21H 和 BIOS INT 17H。
50. 字符码和控制码/功能码。
51. 换页使打印纸到下一页，回车回到行首，换行到下一行。
52. AH=5，DL=字符，INT 21H。
53. DX 指定打印机号 0、1、2。
54. 检测键盘状态，读字符，显示到屏幕，再通过 INT 17H 输出到打印机，满足退出键则结束。
55. 08H 是系统定时器硬件中断，BIOS 08H 程序中调用 INT 1CH 留给用户扩展。
56. 为应用程序留下软接口，使用户可通过改写 1CH 向量实现周期性工作。
57. 每次 1CH 被调用时计数，约 18 次为 1 秒，到时读实时时钟并显示。
58. 保存原 1CH 向量，设置新 1CH 向量，主程序结束前恢复原向量。
59. 中断会打断原程序，必须保护现场；IRET 才能恢复 FLAGS、CS、IP。
60. 分别取 BCD 高 4 位和低 4 位，加 30H 转成对应 ASCII 数字。