final 复习

1 概述

A Von Noyman 结构的组成

运算器

控制器

存储器

输入设备

输出设备

A 有符号数的表示方法与补码

正数不变

负数高位不变其余取反然后+1

2 计算机系统的基本结构与工作原理

B 微程序设计思想

CISC

电路->微指令->微操作->指令(微程序)

RISC

电路->指令

B 存储器分级设计思想

寄存器->高速缓存->内存->外存

C 大端小端格式与字长字的对齐

小端高位高地址

大端高位低地址

A 模型机指令执行流程

微指令->机器指令->宏指令

若干个 CPU 周期->指令周期

若干个时钟周期->总线周期

具体操作

PC 的指令地址数据送入地址缓冲器

数据被译码器译码后经地址总线送入内存单元寻址指令

PC 后移指向下一指令地址

指令操作码送入 IR, 经 ID 译码在 OC 里产生操作控制信号

指令地址码送入地址形成部件, 生成地址信号

根据指令进行操作

C RISC 与 CISC 区别

CISC

指令长度不一

非 load/store 体系

两操作数

move 操作须经寄存器

功能强大, 寻址方式多样, 程序简洁

RISC

指令长度一致

load/store 体系

面向寄存器

精简, 寻址方式简单种类少

三操作数且仅能对寄存器进行运算

指令数量少

程序代码量大

A 流水线原理, 三五级经典划分, 三种冲突解决

指令多级串联会造成部分部件窝工

取指->译码->取操作数->执行->回写

需要在两个指令对应的部件之间加缓冲寄存器来保证并行工作

问题

多指令同周期争用总线

需后一条指令插入气泡阻塞

采用 Harvard 指令解除数据和指令之间的不可同时进行问题

数据冲突

RAW

WAW

WAR

需要优化代码或专用通道定向推送指令来解决

控制冲突

转移指令导致部分流水线中途断流

3 存储器系统

B cache 基本工作原理与作用

基本单元是行或区块

数据字段, 内存中的数据

标志字段, 地址标记寄存器

以区块为单位映射

解决问题

地址映射转换

根据访问不同内容而更新替换 cache

cache 系统基本结构

cache tag

记录数据位置, 是标志字段

判断是否命中

cache 存储体

记录数据内容, 是数据字段

可直接调用片外数据

cache 控制器

决定了 cache 工作效率

主存地址寄存器

存放主存的地址信息, 具体内容取决于地址映射方式

cache 地址寄存器

存放 cache 的地址信息, 具体内容取决于地址映射方式

主存-cache 地址变换部件

建立目录实现转换

替换控制部件

缓存满时按照策略替换数据并更新地址变换部件

映射方法

全相联映像

完全随意对应

cache 分块, 内存分块, 两者块与块对应

直接映像

一对多硬性对应

cache 分块, 内存分页再分块, 两者块与块对应

多路组相联映像

多对多有限随意对应

cache 分组再分块, 内存分页再分块, 两者块与块对应

数据更新

写通方式

同时更新 cache 和主存

写回方式

cachce 被替换掉时写回主存

替换策略

随机替换策略

最不经常使用策略

先进先出策略

近期最少使用策略

cache 脱靶

分块小, 容量小, 替换进的主存块过大或过多

cache 性能

cache 容量, 结构设计优化, 预读取技术

A 存储器的位扩展和字扩展

位扩展在每个存储芯片共享地址线

字扩展在每个存储芯片新增地址线

可以复合扩展

4 总线和接口

B 总线的五种分类方式

位置分类

存储总线

高速连接存储器和 CPU

IO 总线

连接高速外设

扩展总线

连接低速外设

信号传输类型

数据总线

传输数据, 双向三态

字长通常与微处理器字长一致

地址总线(单向)

字长决定了寻址存储单元数量, 乘上单个存储单元内容, 就是寻址空间

控制总线

传输控制信号

电源总线

顾名思义

通信方式

串行总线

一根信号线或两根差分信号线逐个传输数据

并行总线

多根信号线并列在信道上传输

时序控制方式

同步总线

统一时钟

异步总线

各自时钟

时分复用方式

非复用总线

每条信号线功能固定

复用总线

每条信号线不同时间传输信号不同

A 总线周期四个阶段

请求与仲裁阶段

寻址阶段

数据传输阶段

结束阶段

C 常见集中式仲裁与分布式仲裁方法原理优缺点

总线允许, 总线请求, 总线忙

串行集中式仲裁

并行集中式仲裁

串并行混合集中式仲裁

自举分布式仲裁

B 总线时序

同步总线时序

根据最慢的设备统一时间

异步总线时序

需要握手, 不互锁, 半互锁, 互锁

半同步总线时序

对快速设备同步进行, 可以插入时钟周期稍微等待

对慢速设备异步进行, 可以用 ready/wait 决策是否访问

B AHB 数据传输过程

AHB 系统要素

主机

发出操作

从机

承受操作

仲裁器

连接地址和控制选择器

进行决策使用总线的设备

译码器

连接读数据选择器

进行翻译地址

特点

支持突发传输, 可连续传输多数据块

支持分裂式操作, 寻址和数据传输周期分离

单周期总线主机移交, 仲裁器每周期都更新允许信号

单一时钟沿操作, 上升沿

两个阶段

地址阶段, 数据阶段

支持流水线机制

C IO 接口功能, 两种 IO 端口编址方式与特点

功能

设置数据缓冲, 匹配速度

设置电平转换, 匹配电平

设置信息转换, 匹配信息格式

设置时序控制, 匹配时间同步

提供地址译码

提供 IO 控制, 读写控制, 中断控制等

编址方式

存储器映像

IO 端口和存储器统一编址

IO 可看作一个存储单元, 和存储单元统一编址, 无需特别的专用指令

IO 映像

IO 端口和存储器分别编址

IO 单独拿出来, 需要用专门的输入输出指令访问 IO 端口

B 状态查询方式 IO 接口电路原理

程序控制方式

无条件传送

外设永远"准备好"

同步传送方式, 适用于操作简单外设

向主机输入需要三态缓冲器

从主机输出需要锁存器

条件传送

查询传送, 外设许可才可以访问

异步传送方式, 适合大多数外设

向主机输入需要 CPU 查询外设是否 ready

从主机输出需要 CPU 查询外设是否 busy

中断方式

外设发送中断请求, CPU 响应则传送

提高了效率, 外设有一定主动权且可以和 CPU 并行工作

可以适应实时系统对 IO 处理时间的要求

DMA 方式

DMA 控制器临时接管 CPU 的总线, 在 IO 设备和存储器间直接传送

支持多种方式

内存<->外设, 常用

内存<->内存, 少用

IO<->IO, 几乎不用

C I2C 与 SPI 接口原理

I2C

两线式串行总线, 一条数据线一条时钟线

多主机总线, 应答机制

传输过程

开始信号 S

从设备地址

如果是读

发送读命令

收到从机应答

从机发送数据

收到主机应答

继续发送数据.....

如果是写

发送写命令

收到从机应答

主机发送数据

收到从机应答

继续发送数据.....

SPI

同步串行外设接口

单主机总线, 没有应答机制

传输过程

拉低目标从设备电平标识从设备

如果是读

放到 MISO 信号线

如果是写

放到 MOSI 信号线

5 ARM 处理器体系结构和编程模型

C 微架构的概念, Harvard 结构特点, 与 Von Noyman 结构的区别

微架构和 ISA 是并列关系

微架构对应 Von Noyman 和 Harvard 结构

ISA 对应 CISC 和 RISC 处理器

微架构是 ISA 的硬件实现方式

微架构可以不同, 只要 ISA 相同, 则软件兼容

Von Noyman 结构指令和数据同一个总线

Harvard 结构指令和数据分离

简化的 Harvard 结构是对内分离对外同一的指令和数据总线

A M3/4 处理器存储器映射与总线系统

存储器映射

0x00000000

0.5G CODE 区, 代码和重要数据

只能被 Icode 和 Dcode 总线访问

flash 存储代码, SRAM 存储中要数据

考虑成本, 大多数没有 SRAM

0x20000000

0.5G SRAM 区, 数据存储

通过 AHBLite 总线管理

0x4000 0000

0.5G 片上外设区, 用于片上外设

0x60000000

1G 外部 RAM, 用于外部存储器

0xa0000000

1G 片外外设区, 用于片外外设

0xe0000000

0.5G 内核私有区, 用于内核私有外设, 外部私有外设, 厂商定义

总线系统

核心是 AHB 总线协议的内部总线互连矩阵

所连接的各类总线

Icode 总线

CODE 区

取指, 读中断向量

Dcode 总线

CODE 区

数据访问

AHB 系统总线

0x20000000-0xdfffffff

SRAM 区, 片上外设区, 外部 RAM 区, 片外设备区

0xe0100000-0xffffffff

内核私有区的厂商定义

APB/PPB 总线

0xe0040000-0xe00fffff

外部私有外设

C M3/4 处理器两种操作状态, 两种操作模式, 两种访问等级切换

两种操作状态

thumb 状态

执行 thumb 指令的状态

调试状态

处理器被暂停, 通过调试器或进入调试断点后进入

两种操作模式

处理模式

经典处理器的异常模式, 执行中断服务程序

线程模式

除处理模式之外的模式

两种访问等级

特权访问等级

需要系统权限, 有一定危险

非特权访问等级

无需系统权限, 用户即可操作

A M3/4 处理器 16 个常规寄存器与程序状态寄存器 PSR

16 个常规寄存器

13 个通用寄存器

复位之后值是未定义的

R0-R7

低位寄存器, 大多数 16 位 thumb 指令只能访问低位寄存器

R8-R12

高位寄存器, 可用于 32 位指令和少数几个 16 位指令

为实现互相调用, AAPCS 特定标准

R0-R3 用于传参

R4-R11 用于保存局部变量

R12 用于子程序调用中间寄存器

堆栈指针 SP

可用来存两个指针, 但是只有一个可见

选择指针的模式是通过特殊寄存器设定的

大多数情况用 MSP 就够了

R13

主栈指针 MSP, 复位之后或处理模式时使用

进程栈指针 PSP, 只用于线程模式

复位之后是 MSP, 且指向中断向量表第一个字对应地址的内容

链接寄存器 LR

异常处理时用于保存调用返回行为

嵌套调用时需要对 LR 内容压栈保存

最低位无论 0 还是 1, 作为返回地址时为了对齐总认为是 0

R14

程序计数器 PC

可读可写

读的时候每读一次自增 4 指向下一指令地址

写的时候实现程序跳转

很少用写 PC 的功能

R15

特殊寄存器

不映射到内存空间, 只能对名字访问

状态寄存器 PSR

31 30 29 28 27 26-25 24 23-20 19-16 15-10 9 8-0

N Z C V Q ICT/IT T NULL GE ICT/IT NULL 异常/中断编号

解释

N=1

负

Z=1

0

C=1

进位

V=1

溢出

Q=1

饱和

GE

分别对应每个字节数据通路

ICT/IT

指令状态标志位, 用于指令条件执行

T

thumb 指令标志, 总是处于 thumb 状态所以总是 1, 若清零则引起错误异常

异常/中断编号

正在处理的中断编号, 用来决策是否可以抢占嵌套中断

C 堆栈原理, M3/4 的满递减与双堆栈结构

堆栈采用双字对齐

指向栈顶数据, 从高地址向低地址增长

双堆栈

一个特权级, 一个非特权级

特权级一部分用于主栈存储区

非特权级一部分用于进程栈

一般在 SRAM 区, 也可在 CODE 区的 SRAM 里

MSP 在中断向量表第一个元素里, 即 0x00000000

PSP 需要编程时另外给出, 最好用汇编代码

C 位段位带操作

一次存储器操作只访问一个位

SRAM 区低 1M, 0x20000000-0x200fffff

外设区低 1M, 0x40000000-0x400fffff

转换公式

0x42000000+((A-0x40000000)*8+n)*4

原子操作, 不可打断

C 异常处理基本过程与优先级, 优先级分组

支持的异常和中断处理

固定优先级的编号

0, 无异常

1, RST, 优先级-3

复位信号, 优先级最高

2, NMI, 优先级-2

来自 NMI 引脚, 一般由看门狗或掉电检测单元 BOD 产生

3, HardFault, 优先级-1

硬件错误, 异常处理未使能则触发

可编程优先级的编号

4, 存储器错误

访问内存违反 MPU 规则

5, 总线错误

AHB 错误相应

6, 用法错误

无效指令或访问未配置部件

7-10, 保留段

11, SVC

有 OS 时, APP 借此调用系统服务

12, debug 监视

基于软件调试时, 断电和数据观察点等调试事件的异常

13, 保留段

14, PendSV

可挂起的 SVC, 用于 OS 的上下文切换

15, Systick

系统节拍定时器周期异常, 用于定时切换

16-255, IRQ0-240

由片上外设或外设中断源产生

对于 CMSIS 的终端编号, 从-15 开始到 239

配置 NVIC 后

除 RST 外的所有中断都可屏蔽

除 RST, NMI, HardFault 外的所有异常都可单独使能或禁止

除 RST, NMI, HardFault 外的所有异常都可配备比前三者低的优先级

支持优先级动态修改

中断向量表可定位在存储器其他区域, 即支持重定位

低中断处理延迟

处理方式

中断响应时, 直接计算中断类型号*4, 就是中断向量在中断向量表中地址

中断响应时, Icode 总线取指, Dcode 或系统总线同时压栈

中断向量表由 Icode 管理

压栈操作可通过 Dcode 或系统总线管理, 但为简化 CODE 区 SRAM, 只能系统总线完成

系统节拍定时器 Systick

内部一个 24 位递减计数器和四个寄存器

状态控制寄存器 STCSR

加载值寄存器 STRVR

当前计数值寄存器 STCVR

校准值寄存器 STCR

使能后, 从 STCVR 开始递减, 减到 0 产生异常, 再重复该过程

6 ARM 指令系统

C T32 指令汇编语法

操作码和操作数两部分

根据指令前缀区分不同编码格式

[cond][q][S] , [, oprand]

opcode, 操作码

cond, 条件码

q, 指令宽度

S, 含有则指令会更新 APSR

Rd, 目标操作数, 必须是寄存器

Rn, 第一源操作数寄存器, 必须是寄存器

oprand, 第二源操作数, 可是立即数, 寄存器, 经过偏移量计算的寄存器和立即数

A T32 指令 10 种寻址方式

立即数寻址

#a

a 是立即数

内含在指令中

可用 8bit 循环移位得到的立即数是合法的, 因为受制于指令长度限制

寄存器寻址

Ri

i 是寄存器编号

利用寄存器数值作为操作数

寄存器仅限通用寄存器且不可以是 PC

寄存器间接寻址

[Ri]

i 是寄存器编号

利用寄存器存放的数值作为地址, 根据地址寻找操作数

寄存器移位寻址

Ri, opr #a

i 是寄存器编号

opr 是操作

LSL, 逻辑左移, 正常移动

LSR, 逻辑右移, 正常移动

ASR, 算术右移, 补符号位的移动

ROR, 循环右移, 最低位循环到最高位

RRX, 带扩展的循环右移, 用进位符号充当最低位, 一共 33 位参与循环

a 是立即数

通过寄存器得到操作数, 对操作数移位得到最终结果

寄存器偏移寻址

[Ri, #a]

[Ri, Rj]

i, j 是寄存器编号

a 是立即数

Ri 加上 a 或 Rj 的值作为地址, 根据地址寻找操作数

前变址寻址

[Ri, #a]!

[Ri, Rj]!

i, j 是寄存器编号

a 是立即数

指令执行前, 操作数地址存入 Ri

Ri 加上 a 或 Rj 的值作为地址, 根据地址寻找操作数

后变址寻址

[Ri], #a

[Ri], Rj

i, j 是寄存器编号

a 是立即数

指令执行后, 操作数地址存入 Ri

Ri 加上 a 或 Rj 的值作为地址, 根据地址寻找操作数

多寄存器寻址

opr{mode}Ri[!], registers

opr 是操作

STMFD, 压栈

LDMFD, 取出

mode 是模式

IA, 取操作数后基址寄存器自增

IB, 取操作数前基址寄存器自增

DA, 取操作数后基址寄存器自减

DB, 取操作数前基址寄存器自减

M3/4 仅支持 IA 和 DB

i 是寄存器编号

!代表是否取指后更改 Ri 的地址

registers 是寄存器列表

堆栈寻址

oprFD SP!, registers

opr 是操作

STM, 压栈

LDM, 取出

registers 是寄存器列表

PC 相对寻址

label

[PC, #a]

label 是代码标签

a 是立即数

以 PC 当前值作为基地址, 指令中地址标号为偏移量相加寻址

A 基本指令功能语法

MOV, 寄存器, 立即数

LDR, 内存到寄存器

STR, 寄存器到内存

PUSH, 从内存压栈

POP, 向内存弹栈

ADD, 加法

SUB, 减法

B, 无条件跳转

BL, 跳转并把返回地址保存在 LR, 即 R14 中

条件码

EQ, 等于, Z=1

NE, 不等于, Z=0

CS, 进位, C=1

CC, 不进位, C=0

MI, 负数, N=1

PL, 非负数, N=0

VS, 溢出, V=1

VC, 不溢出, V=0

HI, 无符号大于, C=1 且 Z=0

LS, 无符号不大于, C=0 或 Z=1

GE, 有符号不小于, N=V

LT, 有符号小于, N!=V

GT, 有符号大于, Z=0 且 N=V

LE, 无符号不大于, Z=1 或 N!=V

AL, 无条件执行

7 ARM 程序设计

A 伪指令用法(数据定义伪指令重点)

符号定义伪指令

全局变量

GBLA, 全局数字变量初始化 0

GBLL, 全局逻辑变量初始化 False

GBLS, 全局字符串变量初始化 NULL

局部变量

LCLA, 局部数字变量初始化 0

LCLL, 局部逻辑变量初始化 False

LCLS, 局部字符串变量初始化 NULL

变量赋值

SETA, 数字变量赋值

SETL, 逻辑变量赋值

SETS, 字符串变量赋值

寄存器

RLIST, 寄存器列表名称定义

数据定义伪指令

以下指令可以加 U

DCB, 连续字节存储单元并用指定数据初始化

DCW, 连续半字存储单元并用指定数据初始化

DCD, 连续字存储单元并用指定数据初始化

DCFS, 连续单精度浮点数存储单元并用指定数据初始化

DCFD, 连续双精度浮点数存储单元并用指定数据初始化

DCQ, 连续双字存储单元并用指定数据初始化

SPACE, 分配一段连续存储单元

MAP, 定义结构化内存表首地址

FIELD, 定义结构化内存表数据域

汇编控制伪指令

IF, ELSE, ENDIF, 条件

WHILE, WEND, 循环

MACRO, MEND, 宏定义

MEXIT, 循环中提前退出宏

其他伪指令

AREA, 定义代码段或数据段

EXPORT, GLOBAL, 声明全局标号

IMPORT, 通知编译器使用标号在其他文件中

END, 到达程序结尾

A 完整的汇编程序的读写

```
    AREA RESET，CODE，READONLY  ;只读的代码段，RESET 为段名
ENTRY                       ;程序入口点
```

```
START LDR R0，= Num          ;取 Num 地址赋给 R0
LDR R1，[R0]                 ;取 Num 中内容赋给 R1
ADD R1，#0x9A                ; R1=R1+0x9A
STR R1，[R0]                 ;R1 内容赋给 Num 单元
LDR R0，= Nums               ; ……
LDR R2，[R0]
ADD R2，#0xAB
STR R2，[R0]
LOOP B LOOP                  ;无限循环反复执行
END                          ;段结束
```

A C 程序调用汇编函数以及汇编程序调用 C 函数的编程方法

C 程序调用汇编

C 程序

```
extern void foo(int i);
```

汇编程序

```
    EXPORT foo
foo
```

汇编程序调用 C 函数

汇编程序

```
    IMPORT foo
```

C 程序

```
void foo(int i);
```

8 基于 ARM 微处理器硬件与软件系统设计开发

C 嵌入式系统交叉开发环境

宿主机

目标机

宿主机和目标机的连接

物理连接

逻辑连接

通信协议

C 微处理器最小硬件系统

微处理器, 各部件总线连接

处理器内核

数据存储器

程序存储器

定时器与计数器

中断控制器

信号输入输出设备

外设接口

辅助内容

电源系统

时钟系统

复位系统

下载与调试系统

B STM32 时钟树, 功能, 作用, 意义, 特点等

时钟电路

内部时钟

RC 振荡器提供, 不准确不稳定, 误差 1%

外部主时钟源

外部晶振和负载电容, 精确稳定

用户提供的外部时钟信号, 以用户为定

时钟树

输入时钟

时钟频率分

高速时钟

低速时钟

芯片角度分

内部时钟

外部时钟

因此有五种

高速外部时钟 HSE

系统用

高速内部时钟 HSI

8MHz

片内 RC 振荡器产生

上电后作为初始系统时钟

低速外部时钟 LSE

32kHz

外部晶振提供实时时钟

低速内部时钟 LSI

40kHz

片内 RC 振荡器产生

用于实时时钟和看门狗

锁相环倍频输出 PLL

倍频 2-16 倍, 最大 72MHz

输入源可以是高速时钟以及高速时钟的二分频

系统时钟

通常 72MHz

大部分部件的时钟来源

上电后用 HSI, 初始化结束用 HSE

系统时钟分频得到的其他时钟

若干时钟

A GPIO

输出驱动

多路选择器根据设置决定普通输出还是复用功能

输出控制逻辑和一对互补 mos 管来决定输出模式

输入驱动

不能复用

输入信号送到输入数据寄存器同时也给片上外设

工作模式

普通推挽输出, 较大功率驱动器件

普通开漏输出, 只能输出低电平 0, 若想输出高电平需外接电阻和电源

复用推挽输出, 不仅推挽输出, 还可使用片内外设

复用开漏输出, 不仅开漏输出, 还可使用片内外设

上拉输入, 默认上拉至高电平输入

下拉输入, 默认下拉至低电平输入

浮空输入, 用于不确定高低电平输入

模拟输入, 用于外部模拟信号输入

引脚复用

把某些外设功能映射到某些引脚上

引脚复用重映射

把复用引脚从默认引脚转移到备用引脚上

A TIM

三种计数模式

延时时间=(预设值+1)*(预分频系数+1)/时钟频率

向上计数

基本定时器仅有的功能

从 0 开始累加计数, 等到值等于预设值-1, 则溢出

向下计数

从预设值开始累加计数, 等到值等于 1, 则溢出

双向计数

从预设值开始递减计数, 等到值等于 1, 则溢出

然后从 0 开始累加计数, 等到值等于预设值-1, 则溢出

普通输入捕获

用来测量输入信号脉宽或频率

输入的边沿信号发生跳变

利用中断把定时器值存到捕获/比较寄存器中

比较两次捕获的值的差

PWM 输入捕获

将同一输入映射到两个输入信号

其中一个捕获上升沿, 另一个捕获下降沿

利用中断把定时器值存到捕获/比较寄存器中

比较两次捕获的值的差

比较输出

当 CNT 计数值与捕获寄存器 CCR 相等时, 引脚根据事先规定输出信号

PWM 输出

CNT 不停计数, 预设一个 CCR 是占空比

CNT<CCR, 输出一种信号

CNT>=CCR, 输出另一种信号

循环往复

A EXTI

中断优先级

优先级分组

抢占优先级, 主优先级, 决定是否嵌套

子优先级, 从优先级, 决定先后

向量表

默认在 0x00000000, 存储中断服务函数的指针

服务函数 ISR

一般无参数和返回值, 中断发生自动调用执行

每个中断都有对应的服务函数

在启动代码中预定义

除复位函数外, 均预置弱定义属性, 便于其他文件同名函数替代

无需保护和恢复现场, 因为在处理过程中已经自动完成

设置过程

建立向量表

分配栈空间并初始化

设置优先级

设置优先级分组位数

设置抢占优先级和子优先级

使能中断

编写服务函数代码

A USART

串行接口, 支持同步异步

数据收发

发送

内核指令或 DMA 外设, 内存->TDR

发送控制器, TDR->发送移位寄存器->Tx 引脚

TDR 到发送移位寄存器转移完毕, 产生 TDR 空事件 TXE

发送移位寄存器到 Tx 转移完毕, 产生发送完成事件 TC

可在状态寄存器 SR 查询

接收

接收控制器, Rx 引脚->接收移位寄存器->RDR

内核指令或 DMA 外设, RDR->内存

接收移位寄存器到 RDR 转移完毕, 产生接收非空事件 RXNE