EmmmCS Document/HandBook

EasyAI Chen(easyai@outlook.com)
Forewing Xie(jujianai@hotmail.com)
Niffle Wu(1102365274@qq.com)

2018 年 12 月 21 日

目录

第一章　序言

本项目旨在 DE-10 Standard 开发板上，使用 risc-v 架构实现 cpu，并在此基础上实现软件运行环境，对该环境提供 DE-10 的 LED,VGA,PS/2,WM8731 等硬件的驱动并在此环境上开发软件。

第一部分

硬件

第二章　通用模块

2.1　reset_module

2.1.1　基本信息

模块名：reset_module

2.1.2　接口

类型	位宽	名称	说明
input	1	clk	时钟
output	1	rst_n	复位信号（低电平有效）

2.1.3　说明

本模块定义复位模块，用于电路开机自启动、初始化。
在 FPGA 完成开机初始化后复位信号有效，经过 65536 个时钟周期，复位信号无效。

2.2　clkgen_module

2.2.1　基本信息

模块名：clkgen_module

2.2.2 接口

类型	位宽	名称	说明
input	1	clkin	50MHz 时钟
input	1	rst	复位信号
input	1	clken	使能信号
output	1	clkout	输出时钟信号

2.2.3 说明

本模块可产生 50MHz 的可分时钟信号。

第三章　CPU 综述

本项目 CPU 部分采用 RISC-V ISA，依据 riscv-spec-v 2.2 构建。

3.1　指令集支持情况

指令集	版本	支持?
RV32I	2.0	是
RV32E	1.9	是
RV64I	2.0	否
RV128I	1.7	否
Ext.M	2.0	是
Ext.A	2.0	否
Ext.F	2.0	否
Ext.D	2.0	否
Ext.Q	2.0	否
Ext.L	0.0	否
Ext.C	2.0	否
Ext.B	0.0	否
Ext.J	0.0	否
Ext.T	0.0	否
Ext.P	0.1	否
Ext.V	0.2	否
Ext.N	1.1	否

3.2 常量定义

常量均定义在 $rtl/core/cpu\_define.v$ 内，该文件给出 CPU 各个数据位宽和各个预定义模式。

名称	值	说明
CPU_XLEN	32	CPU 字长
REG		
CPU_GREGIDX_WIDTH	5	通用寄存器访问下标位宽
CPU_GREG_COUNT	32	通用寄存器数量
INSTR Decoder		
CPU_INSTR_LENGTH	32	指令长度
CPU_INSTR_INFO_WIDTH	5	指令信息位宽
CPU_INSTR_OPR_INFO_WIDTH	8	指令操作数信息位宽
CPU_INSTR_ DECODE_INFO_WIDTH	13	指令解码信息位宽
CPU_INSTR_OPR_INFO_WIDTH		
CPU_INSTR_OPR_INVALID	b0	无效操作数
CPU_INSTR_OPR_IMM	b00001	操作数：立即数
CPU_INSTR_OPR_RS1	b00010	操作数：RS1
CPU_INSTR_OPR_RS2	b00100	操作数：RS2
CPU_INSTR_OPR_RD	b01000	操作数：RD
CPU_INSTR_OPR_RS3	b10000	操作数：R
CPU_INSTR_GRP_INVALID	0	无效指令组
RV32I		
CPU_INSTR_INFO_WIDTH		
CPU_INSTR_GRP_LUI	1	指令组：LUI
CPU_INSTR_GRP_AUIPC	2	指令组：AUIPC
CPU_INSTR_GRP_JAL	3	指令组：JAL
CPU_INSTR_GRP_JALR	4	指令组：JALR
CPU_INSTR_GRP_BCC	5	指令组：BCC(BEQ...)
CPU_INSTR_GRP_LOAD	6	指令组：LOAD
CPU_INSTR_GRP_STORE	7	指令组：STORE
CPU_INSTR_GRP_ALUI	8	指令组：ALUI
CPU_INSTR_GRP_ALU	9	指令组：ALU
CPU_INSTR_GRP_FENCE	10	指令组：FENCE
CPU_INSTR_GRP_E_CSR	11	指令组：ECALL,EBREAK,CSR
[M]		
CPU_INSTR_GRP_MULDIV	12	指令组：MULDIV
[F]		
CPU_INSTR_GRP_F_FLW	13	指令组：FLW

第四章　CPU

各模块位于同名文件内。

4.1　cpu_gregs

4.1.1　基本信息

模块名：cpu_gregs

4.1.2　接口

类型	位宽	名称	说明
input	1	clk	时钟
input	1	rd_wen	目标寄存器写使能
input	1	reset_n	复位信号
input	CPU_GREGIDX_WIDTH	rs1_idx	rs1 下标
input	CPU_GREGIDX_WIDTH	rs2_idx	rs2 下标
input	CPU_GREGIDX_WIDTH	rd_idx	rd 下标
output	CPU_XLEN	rs1_dat	rs1 数据
output	CPU_XLEN	rs2_dat	rs2 数据
input	CPU_XLEN	rd_dat	rd 数据

4.1.3　说明

本模块定义 CPU 通用寄存器组，在时钟上升沿完成数据写入和读取，遵循先读后写顺序。

4.2 cpu_alu

4.2.1 基本信息

模块名：cpu_alu
所需时间：1 到 32 个周期

4.2.2 宏定义

alu_define.v		
名称	值	说明
ALU_RUN	0	ALU 工作状态码
ALU_STOP	1	ALU 停止状态码
ALU_WIDTH	4	ALU 功能选择端位宽
ALU_ADD	b'0000	加
ALU_SUB	b'0001	减
ALU_AND	b'0010	逻辑与
ALU_OR	b'0011	逻辑或
ALU_XOR	b'0100	逻辑异或
ALU_SLL	b'0101	左移
ALU_SRL	b'0110	逻辑右移
ALU_SRA	b'0111	算术右移
ALU_MUL	b'1000	乘
ALU_MULU	b'1001	无符号乘
ALU_MULSU	b'1010	左操作数有符号，右操作数无符号乘
ALU_DIV	b'1011	除
ALU_DIVU	b'1100	无符号除
ALU_REM	b'1101	取模
ALU_REMU	b'1110	无符号取模
ALU_NOP	b'1111	无操作

4.2.3 接口

类型	位宽	名称	说明
input	1	clk	时钟信号
input	32	src_A	A 输入端
input	32	src_B	B 输入端
input	4	select	功能控制端
input	1	RST	复位端
output	1	READY	预备信号
output	32	dest	输出端
output	4	flags	{CF, OF, ZF, SF}

4.2.4 select 控制端真值表

select	指令	操作
0000	ADD	$dest := src\_A + src\_B$
0001	SUB	$dest := src\_A - src\_B$
0010	AND	$dest := src\_A \ \& \ src\_B$
0011	OR	$dest := src\_A \mid src\_B$
0100	XOR	$dest := src\_A \oplus src\_B$
0101	SLL	$dest := src\_A \ll src\_B$
0110	SRL	$dest := src\_A \gg src\_B$
0111	SRA	$dest := src\_A \gg src\_B$
1000	MUL	$dest := src\_A * src\_B$
1001	MULU	$dest := src\_A * src\_B$
1010	MULSU	$dest := src\_A * src\_B$
1011	DIV	$dest := src\_A / src\_B$
1100	DIVU	$dest := src\_A / src\_B$
1101	REM	$dest := src\_A \ \% \ src\_B$
1110	REMU	$dest := src\_A \ \% \ src\_B$
1111	NOP	$dest := 0$

4.2.5 说明

RST 复位端高电平有效，乘法、除法、取余操作必须透过 RST 控制。进行乘除取余运算前，必须依序给出 RST 的上升沿和下降沿，重置 ALU 后，ALU 才会开始进行运算，直到预备信号 READY 为高电位，表示运算完成。
乘法操作 (MUL, MULU, MULSU) 需要 5 个时钟周期。除法和取余操作需要 32 个时钟周期。其余指令可视为组合电路，可以一个周期内完成操作
标志位仅在加减法时有效，并且按 i386 手册上定义设置。
指令带 U 后缀表示操作为无符号整形操作。若无后缀，默认为有符号整形操作。注意，指令 MULSU 为 $sign * unsigned$ 操作。

4.3 cpu_bus

4.3.1 基本信息

模块名：cpu_bus
所需时钟周期：1 到 5 个周期

4.3.2 宏定义

bus_define.v		
名称	值	说明
BUS_RUN	1'b0	BUS 工作状态码
BUS_STOP	1'b1	BUS 停止状态码
BUS_READ_32	2'b00	读出 32 位数据
BUS_WRITE_8	2'b01	写入 8 位数据
BUS_WRITE_16	2'b10	写入 16 位数据
BUS_WRITE_32	2'b11	写入 32 位数据

4.3.3 内部接口

类型	位宽	名称	说明
input	1	clk	时钟信号
input	32	address	地址端
input	32	wdata	写入数据
input	2	WLEN	SDRAM 读写控制
input	1	EN_N	使能信号 (低有效)
output	1	READY	预备信号
output	32	rdata	读取数据

4.3.4 WLEN 控制信号真值表

WLEN[1]	WLEN[0]	功能
0	0	32 位读取操作
0	1	8 位写入操作
1	0	16 位写入操作
1	1	32 位写入操作

4.3.5 说明

SDRAM 接口部份，请参考 DE10-Standard User Manual 第 39 页 Table3-23。将复位端 RST 置零后，总线开始进行读写操作。当 READY 恢复高电平时，表示读写操作已完成。因为没有输入数据缓存，操作完成之前更改输入可能发生无法预料的结果。
地址端输入映射存储位置详细请见第四章硬件驱动说明。除了 SDRAM 外，其余位于片内存储器的读写所需周期皆为 1 个周期，READY 直接置高电平。
SDRAM IS DEPRECATED

4.4 cpu_instr_decoder

4.4.1 基本信息

模块名：cpu_instr_decoder

4.4.2 接口

类型	位宽	名称	说明
input	CPU_INSTR_LENGTH	instr	输入指令
output	CPU_GREGIDX_WIDTH	rs1_idx	rs1 下标
output	CPU_GREGIDX_WIDTH	rs2_idx	rs2 下标
output	CPU_GREGIDX_WIDTH	rs3_idx	rs3 下标
output	CPU_GREGIDX_WIDTH	rd_idx	rd 下标
output	CPU_XLEN	imm	立即数
output	FUNCT_WIDTH	funct	funct
output	OPCODE_WIDTH	opcode	opcode
output	CPU_INSTR_DECODE_INFO_WIDTH	dec_instr_info	指令解码信息
output	1	instr_valid	指令是否有效
output	1	fp_rm	浮点数 rm
output	1	fp_width	浮点数 width
output	1	fp_fmt	浮点数 fmt

4.4.3 常量

名称	值	说明
FUNCT_WIDTH	10	funct 位宽
OPCODE_WIDTH	7	opcode 位宽

4.4.4 说明

本模块定义 CPU 指令解码器，对输入指令给出相应解码内容。应注意指令解码信息 (dec_instr_info) 中，低 CPU_INSTR_OPR_INFO_WIDTH 位为操作数信息，高 CPU_INSTR_INFO_WIDTH 位为指令组信息。具体信息编码见 1.2 节。

4.5 cpu

4.5.1 基本信息

模块名：cpu

功能：CPU 顶层模块

4.5.2 接口

类型	位宽	名称	说明
input	1	clk	硬件时钟信号
input	1	clr_n	清零信号（低电平有效）
output	1	cpu_clk	CPU 时钟

4.5.3 时序

$$T(\mathit{cpu\_clk}) = 10T(\mathit{clk}) + \mathsf{max}(T(\mathit{cpu\_bus}) + T(\mathit{cpu\_alu}))$$

4.5.4 说明

本模块定义 CPU 顶层模块，用于综合各 CPU 组件并完成指令执行。clk 与 clr_n 信号含义如说明；cpu_clk 为 CPU 时钟信号，每一个周期表明一个 CPU 时钟周期。

第五章　CPU 中断、异常

5.1　综述

本项目中 CPU 中断异常处理在 RISC-V 架构上有所简化，并且进行了修改，请软件开发者遵循本文档所约定的方法进行处理。

5.2　中断控制

初始化

通过写入 mtvec 寄存器，设置中断处理向量表。

打开中断

通过给 mie 寄存器赋值为 1，打开中断。

关闭中断

通过给 mie 寄存器赋值为 0，关闭中断。

中断返回

通过 MRET 指令，从中断中返回。

5.3 中断发生

当中断发生时，CPU 将会储存 x1 到 x31 寄存器、pc 寄存器的一份拷贝。并跳转到 mtvec[0] 所指向的中断处理函数。其中，mcause 寄存器存储中断号，mscratch 存储第一个参数，如硬件中断需要更多的参数传递，由 mhpmevent3 至 mhpmevent31 寄存器存储。

5.4 中断返回

由于本项目仅支持 M-level，故只支持通过 MRET 指令进行中断返回；执行本指令时，将会将 pc、xi 等通用寄存器恢复为进行中断前的状态。

5.5 中断表

中断号	含义
0	FATAL ERROR
1	键盘
2	计时器

5.6 硬件中断参数

5.6.1 键盘

参数 1：键盘扫描码

第六章　硬件驱动综述

本章描述软件与硬件间操作接口。按照 RISC-V 要求，所有地址按四字节对齐。注意开发板可用存储空间为 679KB。

6.1 内存

软件可见内存为 512KB，内存地址为 0H-7 ffffH。

6.2 LED

LED 使用 2 字节大小的 reg 进行保存，取低 10 位作为 LED 控制位。OS 可见的内存地址映射为 8 0000H-8 0001H。

6.3 VGA

VGA 内部需要存储相应字模，本部分为只读内容。VGA 将暴露自身的显存缓冲区和控制寄存器给 OS。VGA 工作模式应该能支持它每行显示不少于 80 个字符，整屏幕显示不少于 30 行。VGA 显存缓冲区要求其存储至少为 30X80X16bit。OS 可见的内存地址映射为：显存缓冲区：8 0004H-8 12C3H。预留 16 字节作为 VGA 控制寄存器区域，内存地址映射为 8 12C4H-8 12D4H。

6.4 键盘

6.5 WM8731

WM8731 IS DEPRECATED

6.6 常量定义

常量均定义在 $rtl/driver/driver\_define.v$ 内

名称	值	说明
LED		
LED_REG_WIDTH	32	LED 寄存器位宽
DISPLAY/VGA		
DP_X_ADDR_WIDTH	8	水平地址位宽
DP_Y_ADDR_WIDTH	5	垂直地址位宽
DP_REG_WIDTH	128	显示控制寄存器位宽
DP_COLOR_WIDTH	4	显示颜色位宽

第七章　硬件驱动

本章提供各硬件驱动的说明。

7.1　led 驱动

7.1.1　基本信息

模块名：led_ctrl

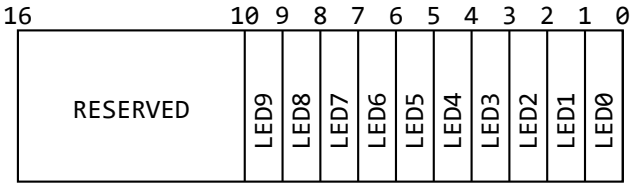
7.1.2　接口

类型	位宽	名称	说明
input	LED_REG_WIDTH	led_reg	led 控制寄存器
硬件控制信号			
output	1	led_0	led0
output	1	led_1	led1
output	1	led_2	led2
output	1	led_3	led3
output	1	led_4	led4
output	1	led_5	led5
output	1	led_6	led6
output	1	led_7	led7
output	1	led_8	led8
output	1	led_9	led9

7.1.3 说明

本模块提供对 led 的硬件驱动，OS 可通过 LED 控制寄存器的内存映射直接控制 led，注意控制寄存器中只有低 10 位有定义，高位保留。

7.1.4 控制寄存器



7.2 显示驱动

7.2.1 VGA 驱动

基本信息

模块名：vga_ctrl（内部模块）

接口

类型	位宽	名称	说明
input	1	pclk	25MHz 时钟
input	1	reset	置位
input	24	vga_data	vga 数据
output	10	h_addr	水平方向扫描像素点坐标
output	10	v_addr	垂直方向扫描像素点坐标
output	1	hsync	行同步信号
output	1	vsync	列同步信号
output	1	valid	消隐信号
output	8	vga_r	R 颜色信号
output	8	vga_g	G 颜色信号
output	8	vga_b	B 颜色信号

说明

本模块提供底层 VGA 控制。

7.2.2 显示驱动

基本信息

模块名：display_ctrl

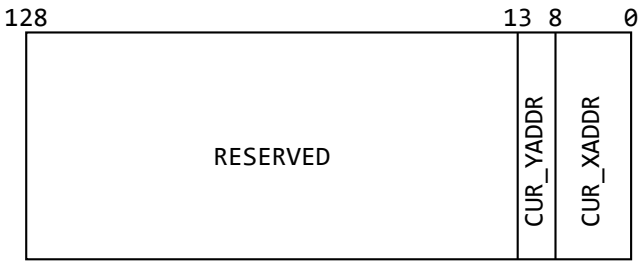
接口

类型	位宽	名称	说明
input	1	clk	50MHz 时钟
input	1	reset_n	复位
input	16	in_data	输入数据
input	DP_REG_WIDTH	ctrl_reg	控制寄存器
output	DP_X_ADDR_WIDTH	x_addr	水平坐标
output	DP_Y_ADDR_WIDTH	y_addr	垂直坐标
硬件控制信号			
input	1	vga_rst	VGA 复位信号
output	1	vga_vs	VGA 列同步信号
output	1	vga_hs	VGA 行同步信号
output	1	vga_syncn	VGA 同步信号
output	1	vga_blankn	VGA 消隐信号
output	1	vga_clk	VGA 时钟信号
output	8	vga_r	VGA R 颜色信号
output	8	vga_g	VGA G 颜色信号
output	8	vga_b	VGA B 颜色信号

说明

本模块提供显示控制驱动。输入数据 in_data 中，低 8 位为 ASCII 码，高 8 位为颜色数据，其中，高 4 位为前景色，低 4 位为背景色。

控制寄存器



其中，CUR_XADDR 和 CUR_YADDR 代表当前光标位置。

颜色表

名称	值	说明
颜色代码		
VGA_BLACK	0x0	黑
VGA_BLUE	0x1	蓝
VGA_GREEN	0x2	绿
VGA_CYAN	0x3	青
VGA_RED	0x4	红
VGA_MAGENTA	0x5	洋红
VGA_BROWN	0x6	棕
VGA_WHITE	0x7	白
VGA_YELLOW	0xe	黄
对应 RGB		
VGA_RGB_BLACK	000000	黑
VGA_RGB_BLUE	0000ff	蓝
VGA_RGB_GREEN	008000	绿
VGA_RGB_CYAN	00ffff	青
VGA_RGB_RED	ff0000	红
VGA_RGB_MAGENTA	ff00ff	洋红
VGA_RGB_BROWN	a52a2a	棕
VGA_RGB_WHITE	ffffff	白
VGA_RGB_YELLOW	ffff00	黄

字符扩展

编码	说明
0x1	实心方框
0x2	上半实心方框
0x3	下半实心方框
0x4	实心圆
0x5	上三角形
0x6	右三角形
0x7	右上三角形
0xe	右下三角形
0xf	左三角形
0x10	左上三角形
0x11	左下三角形
0x12	下三角形

第二部分

软件

第八章　Kernel 综述

8.1　架构

本 Kernel 不提供进程、线程模型，文件系统，网络。
本 Kernel 由以下部分组成：

- KLIB - Kernel 编程的基础库
- driver - 提供对 VGA、PS/2 KB、LED 等设备控制
- intr - 中断处理
- mm - 内存管理器，提供动态内存管理
- sh - 提供交互式命令行
- app - 内置应用

8.2　常量、类型

名称	值	说明
NULL	0	NULL 值
FALSE	0	布尔 False
TRUE	1	布尔 True

名称	对应类型	说明
BOOL	unsigned char	布尔类型
u8	unsigned char	8 位无符号数
u16	unsigned short	16 位无符号数
u32	unsigned int	32 位无符号数
s8	char	8 位有符号数
s16	short	16 位有符号数
s32	int	32 位有符号数

第九章　Kernel 中断处理

中断规定、中断表见 CPU 中断部分。**注意 Kernel 统一将中断处理函数设置为 intr，通过在 intr 内部读取中断号进行中断分发，不再由 CPU 进行中断分发**

9.1　intr_init

`void intr_init(void)`

简介：对中断模块进行初始化。

9.1.1　函数功能

开机时被调用，初始化中断模块。

9.2　intr_open

`void intr_open(void)`

简介：打开中断。

9.3　intr_close

`void intr_close(void)`

简介：关闭中断。

9.4 intr

`void intr(void)`
简介：中断处理主函数，用于分发中断。

9.5 intr_handler_register

`void intr_handler_register(u8 intrno, void *handler)`
简介：对相应中断处理函数进行注册。

第十章 Kernel 驱动

10.1 LED

LED 使用 2 个字节的控制寄存器，内存映射在 0x80000 起始。

10.1.1 接口

led_init

函数原型：void led_init(void);
说明：led 驱动初始化

led_on

函数原型：void led_on(u8 id);
参数：

类型	名称	说明
u8	id	取值 0-9，对应 led 的 id

说明：开对应 led，若 id 非法，无操作。

led_off

函数原型：void led_off(u8 id);
参数：

类型	名称	说明
u8	id	取值 0-9，对应 led 的 id

说明：关对应 led，若 id 非法，无操作。

led_toggle

函数原型：`void led_toggle(u8 id);`
参数：

类型	名称	说明
u8	id	取值 0-9，对应 led 的 id

说明：反转对应 led 状态，若 id 非法，无操作。

10.2 VGA

屏幕坐标，以左上角为原点，横向为 x 轴，纵向为 y 轴。

10.2.1 常量、宏、类型

名称	值	说明
VGA_CHAR_X_SIZE	80	列数
VGA_CHAR_Y_SIZE	30	行数
VGA_CHAR_BUF_SIZE	2400	总字符数

名称	值	说明
背景色		
VGA_B_BLACK	0x00	黑
VGA_B_BLUE	0x10	蓝
VGA_B_GREEN	0x20	绿
VGA_B_CYAN	0x30	青
VGA_B_RED	0x40	红
VGA_B_MAGENTA	0x50	洋红
VGA_B_BROWN	0x60	棕
VGA_B_WHITE	0x70	白
前景色		
VGA_F_BLACK	0x00	黑
VGA_F_BLUE	0x01	蓝
VGA_F_GREEN	0x02	绿
VGA_F_CYAN	0x03	青
VGA_F_RED	0x04	红
VGA_F_MAGENTA	0x05	洋红
VGA_F_BROWN	0x06	棕
VGA_F_WHITE	0x07	白
VGA_F_YELLOW	0x0e	黄
数字格式		
VGA_N_S_DEC	0x0	有符号十进制
VGA_N_U_DEC	0x1	无符号十进制
VGA_N_HEX	0x2	十六进制

10.2.2 接口

vga_init

函数原型：`void vga_init(void);`
说明：vga 驱动初始化

vga_writec

函数原型：`void vga_writec(u8 color, char c, u8 x, u8 y);`
参数：

类型	名称	说明
u8	color	颜色
char	c	输出字符
u8	x	x 坐标
u8	y	y 坐标

说明：在对应坐标处写入字符，若坐标非法则无操作。注意该函数不改变光标位置。

vga_putc

函数原型：`void vga_putc(u8 color, char c);`
参数：

类型	名称	说明
u8	color	颜色
char	c	输出字符

说明：在光标处写入字符，该函数会自动滚屏。

vga_puts

函数原型：`void vga_puts(u8 color, const char *str);`
参数：

类型	名称	说明
u8	color	颜色
const char*	str	输出字符串

说明：在光标处写入字符串，该函数会自动滚屏。

vga_putn

函数原型：`void vga_putn(u8 color, u32 n, u8 mode);`
参数：

类型	名称	说明
u8	color	颜色
u32	n	输出数字
u8	mode	数字格式

说明：在光标处写入数字，该函数会自动滚屏。

vga_clean

函数原型：`void vga_clean(void);`
说明：清屏

10.3 键盘

KBD 维护一个环形字符缓冲区。

10.3.1 接口

kbd_update

函数原型：`void kbd_update(u8* args);`

使用一个两字节的内存空间维护键盘状态，cpu 在中断前应写好对应地址内容

参数	含义
args[0]	是否有效
args[1]	键盘扫描码

说明：中断处理函数，在键盘按下时被调用；作用是把键盘操作写入缓冲区。

kbd_getc

函数原型：`u8 kbd_getc();`
说明：从缓冲区读取一位字符 ascii 码并返回。

第十一章 Kernel Lib

若函数与 C 标准库中函数同名，则其行为也一致。该部分函数文档见 C11 标准文档。

11.1 DList

提供了侵入式双向链表。链表头结点不存放数据。

11.1.1 接口

DLIST_INIT

函数原型：DLIST_INIT(list, m_next, m_prev)
参数：

类型	名称	说明
ptr	list	链表头结点
member name	m_next	后驱变量名
member name	m_prev	前驱变量名

说明：初始化链表

DLIST_EMPTY

函数原型：DLIST_EMPTY(list, m_next, m_prev)
参数：

类型	名称	说明
ptr	list	链表头结点
member name	m_next	后驱变量名
member name	m_prev	前驱变量名

说明：判断链表是否为空

DLIST_INSERT

函数原型：DLIST_INSERT(pos, m_next, m_prev, node)
参数：

类型	名称	说明
ptr	pos	被插入结点的前驱
member name	m_next	后驱变量名
member name	m_prev	前驱变量名
ptr	node	被插入结点

说明：插入结点

DLIST_DELETE

函数原型：DLIST_DELETE(list, pos, m_next, m_prev)
参数：

类型	名称	说明
ptr	list	链表
ptr	pos	被删除结点
member name	m_next	后驱变量名
member name	m_prev	前驱变量名

说明：删除结点

DLIST_ADD_TAIL

函数原型：DLIST_ADD_TAIL(list, m_next, m_prev, node)
参数：

类型	名称	说明
ptr	list	链表
member name	m_next	后驱变量名
member name	m_prev	前驱变量名
ptr	node	被添加结点

说明：添加结点至链表尾

DLIST_ADD_HEAD

函数原型：DLIST_ADD_HEAD(list, m_next, m_prev, node)
参数：

类型	名称	说明
ptr	list	链表
member name	m_next	后驱变量名
member name	m_prev	前驱变量名
ptr	node	被添加结点

说明：添加结点至链表头

DLIST_FOREACH

函数原型：DLIST_FOREACH(list, m_next, ptr)
参数：

类型	名称	说明
ptr	list	链表
member name	m_next	后驱变量名
ptr	ptr	当前操作结点

说明：链表遍历

DLIST_FIND_NODE

函数原型：DLIST_FIND_NODE(list, m_next, p_node, key, cmpfunc)
参数：

类型	名称	说明
ptr	list	链表
member name	m_next	后驱变量名
ptr	p_node	返回值
value	key	关键字
function	cmpfunc	比较函数

比较函数要求：二元函数，左操作数接受 key 值，右操作数接受一个指向链表结点的指针，若匹配返回非零值。
说明：链表结点查找，若查找不到返回 NULL。

11.2 stdio

11.2.1 接口

getchar

函数原型：char getchar(void);
从键盘读取一个字符，返回其 ascii 码。

gets

函数原型：char* gets(char* str);
从键盘读取一个字符串，写入 str 中，以

'\0'

结尾。
返回值为 str。

putchar

函数原型：int putchar(char cha);
将 cha 输出到 vga 缓冲区。

puts

函数原型：int puts(const char* str);
将字符串 str 输出到 vga 缓冲区。

putn

函数原型：`void putn(u32 n, u8 mode);`

参数说明：

类型	名称	说明
u32	n	输出数字
u8	mode	数字格式

putchar_color

函数原型：`int putchar_color(u8 color, char cha);`
将 `cha` 输出到 `vga` 缓冲区，带有颜色 `color`。

puts_color

函数原型：`int puts_color(u8 color, const char * str);`
将字符串 `str` 输出到 `vga` 缓冲区，带有颜色 `color`。

putn_color

函数原型：`void putn_color(u8 color, u32 n, u8 mode);`

类型	名称	说明
u8	color	颜色
u32	n	输出数字
u8	mode	数字格式

第十二章　Kernel 内存管理

Kernel 提供 128KB 的堆内存管理，采用 Buddy Memory Management。本管理器提供 16B,32B,64B,128B,256B,512B,1KB,2KB,4KB,8KB,16KB,32KB,64KB,128KB 共 14 层分配大小。**注意实际分配大小为需求大小 +4B 指针 Header 后向上 2 的幂对齐**

12.1　常量、宏、类型

名称	值	说明
HEAP_SIZE	131072	128KB
TREE_DEPTH	14	分配粒度

buddy_mm_info_t

成员：

类型	名称	说明
buddy_mm_info_t*	m_next	链表，后驱
buddy_mm_info_t*	m_prev	链表，前驱

说明：空闲块链表项。

buddy_mm_header_t

类型：u32
说明：指针 Header，存储 level

buddy_mm_t

成员：

类型	名称	说明
u32	used_size	已使用大小
buddy_mm_info_t*[TREE_DEPTH]	tree	空闲块集合

说明：Buddy 内存分配系统。

12.2 接口

mm_init

函数原型：void mm_init(void);
说明：内存管理初始化。

mm_alloc

函数原型：void* mm_alloc(u32 sz);
参数：

类型	名称	说明
u32	sz	需分配大小

返回值：内存块起始地址，若分配失败返回 NULL
说明：内存块分配。

mm_dealloc

函数原型：void mm_dealloc(void *p);
参数：

类型	名称	说明
void*	p	需回收内存块指针

说明：回收相应内存块。