PAWS a Risc-V RV32IMACB CPU – Programming Guide

About

PAWS is a project to occupy my time in retirement, with the aim of teaching myself about FPGA programming. It is based upon the idea of the 8-bit computers and consoles from the 1980s, but using a modern CPU.

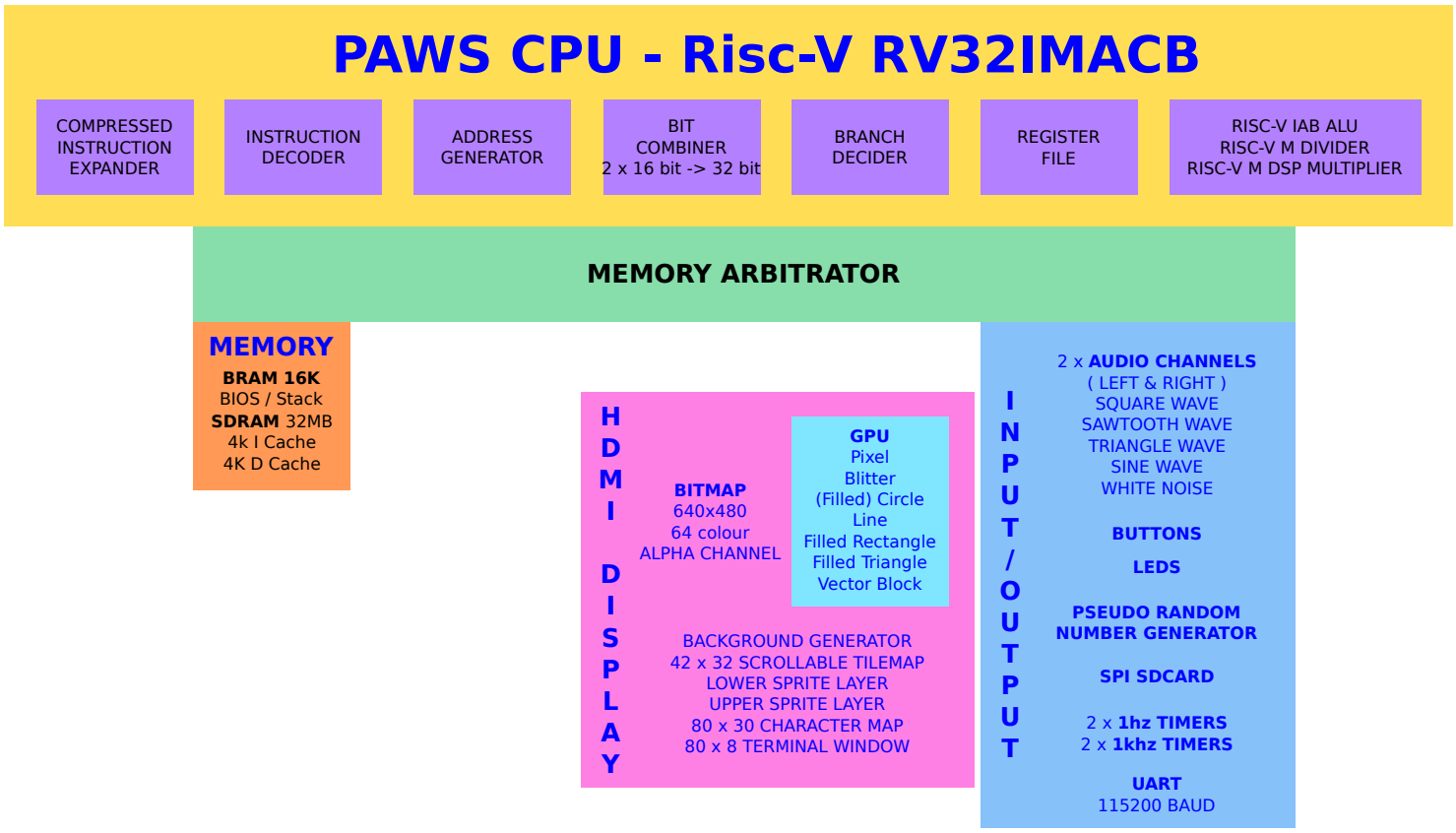
It is a development from the work I did on the J1 CPU, which is a 16-bit CPU designed to run Forth natively, to which I added a display via HDMI, plus various input/out facilities such as a UART, button input, LED output and basic audio.

The J1 CPU has been swapped out for a Risc-V RV32 processor, as this can easily be programmed via C using GCC, in addition to being relatively straightforward to being implemented on an FPGA.

A support library, libPAWS, for easy access to the hardware is provided. This documentation details libPAWS and describes the hardware.

PAWS a Risc-V RV32IMACB CPU – Programming Guide

PAWS CPU and SOC



PAWS BIOS

Upon startup PAWS boots to the BIOS, which initialises the display, clears any input/output buffers, and reads the ROOT DIRECTORY from PARTITION 0 of a FAT16 formatted SDCARD.

PAW (compiled programs) files are display for selection, scrolling through the available files using “FIRE 2” and selecting a file for loading and executing using “FIRE 1”.

Upon selection, the BIOS will load the selected PAW into SDRAM, reset the display, and launch the selected PAW program.

Compiling Programs For PAWS

The default language for PAWS is C, specifically GCC.

To create a program for PAWS, create a C file in the SOFTWARE/c directory. It is advised to use the SOFTWARE/template.c as a starting point.

Contents of template.c	Explanation
`#include "PAWSlibrary.h"`	Use libPAWS for definitions and helper functions.
`void main( void ) {`	MAIN program loop
`    INITIALISEMEMORY();`	Setup the memory map
`    while(1) {`	Main loop.
`    }`	
`}`	

Compile your code using the helper shell script. For example, to compile the included asteroids style arcade game, `./compile_SDRAM.sh c/asteroids.c`. This will compile the program to build/code.PAW, which can be copied to the SDCARD for loading via the BIOS.

Colours

PAWS uses a 6-bit colour attribute, given as RRGGBB. This gives 64 colours, specified in decimal as per the table below. Names defined in libPAWS are given below the decimal representation.

0 BLACK	1	2 DKBLUE	3 BLUE	4	5	6	7
8 DKGREEN	9	10	11 DKCYAN	12 GREEN	13	14	15 CYAN
16	17	18	19 PURPLE	20	21 GREY1	22	23
24	25	26	27	28	29	30	31
32 DKRED	33	34 DKMAGENTA	35	36	37	38	39
40 DKYELLOW	41	42 GREY2	43	44	45	46	47
48 RED	49	50	51 MAGENTA	52	53	54	55
56 ORANGE	57	58	59	60 YELLOW	61	62	63 WHITE

In addition, for layers that allow, tilemap, bitmap and character map, there is a TRANSPARENT attribute, which allows the layer below to show through. See relevant sections of the documentation for further explanation.

Memory Management

The BIOS will initialise the memory, and allocates space at the top of fast BRAM memory for the CPU STACK, and space at the top the SDRAM for SDCARD buffers.

Address Range	Memory Type	Usage
0x00000000 – 0x00004000	Fast BRAM	0x00000000 – 0x00001388 BIOS 0x00004000 – 0x00002000 STACK
0x00008000 – 0x0000ffff	I/O Registers	Commuincation with the PAWS hardware. No direct hardware access is required, as libPAWS provides functions for all aspects of the PAWS hardware.
0x1000000 -0x1fffffff	SDRAM	Program and data storage. Accessed via instruction and data caches. SDCARD buffers and structures are allocated at the top of this address range.

libPAWS variables and functions

`unsigned char *MEMORYTOP`	Points to the top of unallocated memory.
`void INITIALISEMEMORY( void )`	Sets up the memory map using parameters passed from the BIOS. Allocates the buffers used for SDCARD access, and correctly sets MEMORYTOP.
`unsigned char *memoryspace( unsigned int size )`	Returns a pointer to a buffer of at least size bytes, allocated from the top of the free memory space. Aligned to 16-bit address.
`unsigned char *filememoryspace( unsigned int size )`	Returns a pointer to buffer of at least size bytes, allocated from the top of the free memory space. Aligned to 16-bit address. This should be used for preference when allocating memory in which to read a file, as the buffer will contain sufficient space to account for the minimum block size of the SDCARD.

NOTE: memoryspace and filememoryspace are similar to the standard C function malloc. There is no mechanism for freeing memory in libPAWS.