Overview

This custom assembly language is designed with older systems in mind. For instance, although in modern systems graphics are mainly passed in as triangles, and processed by the GPU through different geometry and fragment shaders, older systems displayed pixels using long lists of pixel colors that were passed directly to displays. This was not very good for adjustable and scalable screen sizes, but for very specific use cases, it still should work totally fine. I used this for one of my fun instructions, namely DISP. I also allowed the user to make system calls to play tones for adding basic sound effects. Additionally, I integrated some system calls that are used frequently as their own instructions, like PRNT for print string, and CAST and CCAT to support printing and debugging. Despite the many differences, some of the more basic instructions come directly from MIPS, but with more easily understood opcodes. However, even if some of the instructions were inspired by MIPS, the locations of outputs and the reduced instruction set makes it such that the workflow is completely different. Some instructions additionally place outputs in special registers (like the condition register), which separates tasks allowing the instructions to be smaller. I spent a lot of time considering what instructions would be useful, which ones are necessary, and what registers I would need to support the normal functioning of programs and completion of instructions. I also spent a lot of time optimising the bitfields for the instructions, and got a nice 24 bits, or 3 bytes for each instruction. Ideally, you should be able to do anything with this that you could do in MIPS. I would additionally like to specify that my language should have labels, like in MIPS, and comments, perhaps specified with //. Finally, all of the information about the registers, instructions, and bit fields is all listed below, including example programs.

Registers

| General-purpose: | | Special-purpose: | | |
|---|---|---|---|---|
| R0 | 0000 | TP | 1010 | Temporary register used by operations like swap |
| R1 | 0001 | CL | 1011 | Currentline in the program, like PC pointer, not like executing line |
| R2 | 0010 | ST | 1100 | Stack address |
| R3 | 0011 | PS | 1101 | Register represents a stack data structure with special memory for the processor to store the |
| R4 | 0100 | | | position of the stack pointer at a certain time. Similar to RA in MIPS |
| R5 | 0101 | CN | 1110 | Stores the result of an evaluated condition |
| R6 | 0110 | PK | 1111 | Stores multiple values packed into one 32 bit register value |
| R7 | 0111 | | | |
| R8 | 1000 | | | |
| R9 | 1001 | | | |

Basic Instructions (17x)

RI Type:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| SET | Set Value | SET RD imm | RD ← imm | 0011 | rd | imm | | | |
| INC | Increment | INC RD imm | RD ← RD + imm | 0100 | rd | imm | | | |
| LTI | Less Than Immediate | LTI RS imm | CN ← RS < imm | 0101 | rs | imm | | | |
| GTI | Greater Than Immediate | GTI RS imm | CN ← RS > imm | 0110 | rs | imm | | | |
| EQI | Equivalent To Immediate | EQI RS imm | CN ← RS == imm | 0111 | rs | imm | | | |

R Type:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ADD | Add Registers | ADD RS RT RD | RD ← RS + RT | 0000 | 0000 | rs | rt | rd | 0000 |
| LES | Less Than | LES RS RT | CN ← RS < RT | 0000 | 0001 | rs | rt | 0000 | 0000 |
| GRT | Greater Than | GRT RS RT | CN ← RS > RT | 0000 | 0010 | rs | rt | 0000 | 0000 |
| EQV | Equivalent | EQV RS RT | CN ← RS == RT | 0000 | 0011 | rs | rt | 0000 | 0000 |
| ORR | Or | ORR RS RT | CN ← (RS==1 \|\| RT==1) | 0000 | 0100 | rs | rt | 0000 | 0000 |
| AND | And | AND RS RT | CN ← (RS==1 && RT==1) | 0000 | 0101 | rs | rt | 0000 | 0000 |
| RTN | Return | RTN | CL ← PS.pop() | 0000 | 0110 | 0000 | 0000 | 0000 | 0000 |

I Type:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| JNC | Jump No Condition | JNC imm | CL ← imm | 0000 | 0111 | imm | | | |
| JWC | Jump With Condition | JWC imm | if (CN==1): CL ← imm | 0000 | 1000 | imm | | | |
| CPT | Checkpoint | CPT imm | PS ← CL+imm | 0000 | 1001 | imm | | | |

O Type:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| SAV | Save | SAV RS offset(RT) | (Stack at RT+offset) ← RS | 1000 | rs | rt | offset | | |
| LOD | Load | LOD RD offset(RS) | RD ← (Stack at RS+offset) | 1001 | rs | rd | offset | | |

Special Instructions (11x)

R Type:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| SWAP | Swap Values | SWAP RS RT | RS ↔ RT | 0000 | 1010 | rs | rt | 0000 | 0000 |
| CAST | Cast to String | CAST RD | RD ← ST; (R0.value →"R0.value"→ Stack) | 0000 | 1011 | 0000 | 0000 | rd | 0000 |
| PARS | Parse String | PARS RD | RD ← parseint(null terminated string at RD) | 0000 | 1100 | 0000 | 0000 | rd | 0000 |
| PRNT | Print String | PRNT RS | SYSCALL print string on address in RS | 0000 | 1101 | rs | 0000 | 0000 | 0000 |
| PLAY | Play Tone | PLAY RS | Play tones stored at RS, null terminated | 0000 | 1110 | rs | 0000 | 0000 | 0000 |
| DISP | Display Pixels | DISP RS | Draw raster pixel buffer from RS to null | 0000 | 1111 | rs | 0000 | 0000 | 0000 |
| MFPK | Move from pk | MFPK RD | RD ← PK | 0001 | 0000 | 0000 | 0000 | rd | 0000 |
| TIME | Epoch Time | TIME RD | RD ← (unix epoch time in seconds) | 0001 | 0001 | 0000 | 0000 | rd | 0000 |
| CCAT | Concatenate | CCAT RS RT RD | RD ← ST; (RS(w/o null)+RT → ST) | 0001 | 0010 | rs | rt | rd | 0000 |

I Type:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ALOC | Allocate | ALOC imm | MIPS equivalent ADDIU $sp, $sp, -4*imm | 0001 | 0011 | imm | | | |

O Type:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| PACK | Pack Int | PACK RS RT offset | PK lowest 2n ← RS lowest n, RT lowest n | 1010 | rs | rt | offset | | |

Binary Encoding

RI Type:

| OPCODE | REGISTER | IMMEDIATE |
|---|---|---|
| 4 bits | 4 bits | 16 bits |

I Type:

| OPCODE | SPECIAL | IMMEDIATE |
|---|---|---|
| 4 bits | 4 bits | 16 bits |

R Type:

| OPCODE | SPECIAL | RS | RT | RD | UNUSED |
|---|---|---|---|---|---|
| 4 bits | 4 bits | 4 bits | 4 bits | 4 bits | 4 bits |

O Type:

| OPCODE | RS | REGISTER 2 | OFFSET |
|---|---|---|---|
| 4 bits | 4 bits | 4 bits | 12 bits |