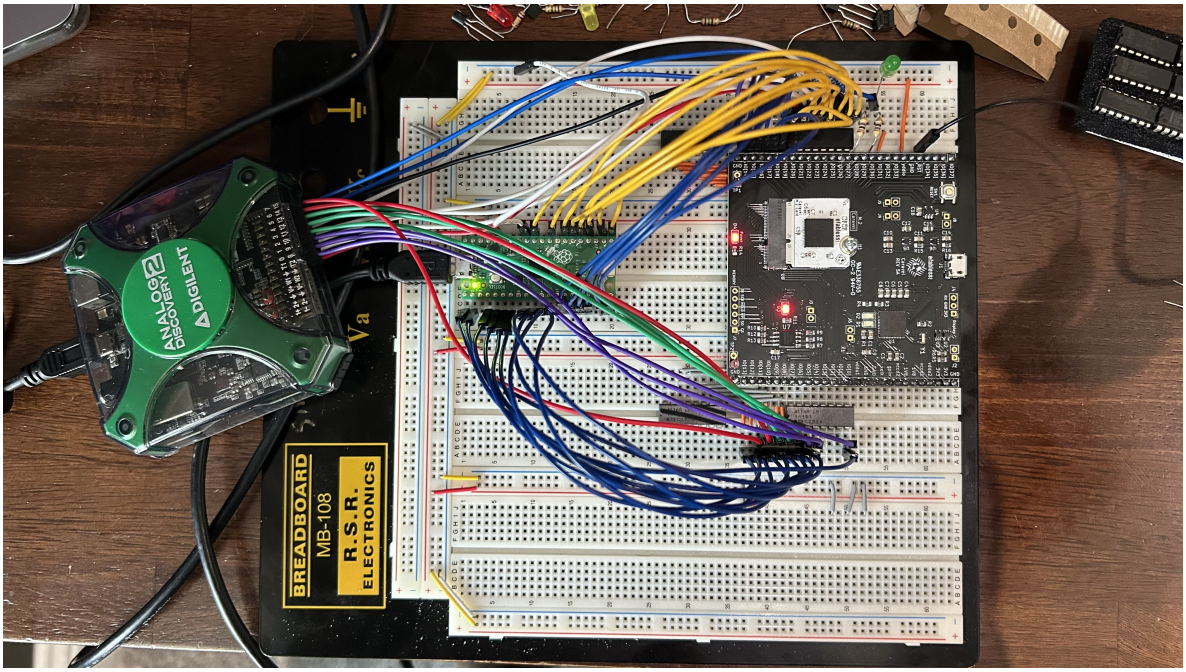
Z23 Manual

Devin Singh
singh956@purdue.edu

May 13, 2024



1 Z23

The Z23 is an accumulator based, 8-bit CPU based off the Zilog Z80 developed during Summer 2023 as part of the Purdue STARS Program. It supports most of the instructions of the Z80 except those using the shadow registers. An instruction table can be seen in Figure 1. More information on the effects of each instruction can be found at this reference guide. The CPU has eight 8-bit registers which can be used in pairs to perform 16-bit memory accesses. There are 8 GPIO pins which can be configured by writing to each pins respective control and access pin. There is support from interrupts from each of the GPIO pins, the dedicated interrupt pin (IO pin 30), and an internal 32 bit timer. Please direct any questions can be directed to me at singh956@purdue.edu.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	nop	ld bc,xx	ld (bc),a	inc bc	inc b	dec b	ld b,x	rlca		add hl,bc	ld a,(bc)	dec bc	inc c	dec c	ld c,x	rrca
1	djnz x	ld de,xx	ld (de),a	inc de	inc d	dec d	ld d,x	rla	jr x	add hl,de	ld a, (de)	dec de	inc e	dec e	ld e,x	rra
2	jr nz,x	ld hl,xx	ld (xx),hl	inc hl	inc h	dec h	ld h,x		jr z,x	add hl,hl	ld hl, (xx)	dec hl	inc l	dec l	ld l,x	cpl
3	jr nc,x	ld sp,xx	ld (xx),a	inc sp	inc (hl)	dec (hl)	ld (hl),x	scf	jr c,x	add hl,sp	ld a, (xx)	dec sp	inc a	dec a	ld a,x	ccf
4	ld b,b	ld b,c	ld b,d	ld b,e	ld b,h	ld b,l	ld b,(hl)	ld b,a	ld c,b	ld c,c	ld c,d	ld c,e	ld c,h	ld c,l	ld c,(hl)	ld c,a
5	ld d,b	ld d,c	ld d,d	ld d,e	ld d,h	ld d,l	ld d,(hl)	ld d,a	ld e,b	ld e,c	ld e,d	ld e,e	ld e,h	ld e,l	ld e,(hl)	ld e,a
6	ld h,b	ld h,c	ld h,d	ld h,e	ld h,h	ld h,l	ld h,(hl)	ld h,a	ld l,b	ld l,c	ld l,d	ld l,e	ld l,h	ld l,l	ld l,(hl)	ld l,a
7	ld (hl),b	ld (hl),c	ld (hl),d	ld (hl),e	ld (hl),h	ld (hl),l	**halt**	ld (hl),a	ld a,b	ld a,c	ld a,d	ld a,e	ld a,h	ld a,l	ld a,(hl)	ld a,a
8	add a,b	add a,c	add a,d	add a,e	add a,h	add a,l	add a, (hl)	add a,a	adc a,b	adc a,c	adc a,d	adc a,e	adc a,h	adc a,l	adc a, (hl)	adc a,a
9	sub b	sub c	sub d	sub e	sub h	sub l	sub (hl)	sub a	sbc a,b	sbc a,c	sbc a,d	sbc a,e	sbc a,h	sbc a,l	sbc a, (hl)	sbc a,a
A	and b	and c	and d	and e	and h	and l	and (hl)	and a	xor b	xor c	xor d	xor e	xor h	xor l	xor (hl)	xor a
B	or b	or c	or d	or e	or h	or l	or (hl)	or a	cp b	cp c	cp d	cp e	cp h	cp l	cp (hl)	cp a
C	ret nz	pop bc	jp nz,xx	jp xx	call nz,xx	push bc	add a,x		ret z	**ret**	jp z,xx	**xxBITxx**	call z,xx	call xx	adc a,x	
D	ret nc	pop de	jp nc,xx		call nc,xx	push de	sub x		ret c		jp c,xx		call c,xx		sbc a,x	
E		pop hl				push hl	and x			jp (hl)					xor x	
F		pop af		di		push af	or x			ld sp,hl		ei		reti	cp x	

Figure 1: Instructions Supported

2 Memory Map

The provided program for the Raspberry Pi Pico provides support for a simple ROM/RAM memory map in addition to the internal memory mapped IO regions. Software wishing to use the stack needs to set the stack pointer to a lower address such as 0x3000 (from its default value of 0xFF00) before pushing/popping data from the stack. The memory map can be seen in Table 1.

Memory Range	Description	Usage
0xFFD0-0xFFD4	Timer Access	A register map consisting of {enable, duration0, duration1, duration2, duration3} bytes.
0xFFC0	GPIO Interrupt Mask	Setting a 1 in each bit will enable interrupts from that respective GPIO pin.
0xFF90-0xFF97	GPIO Pin Access	Read/writes to these addresses will read/write from the GPIO pins.
0xFF80-0xFF87	GPIO Pin Control	Setting a byte to 1 will set the corresponding GPIO pin to output.
0x2000-0x7FFF	RAM	Random access memory used for things like the stack and any R/W memory.
0x0000-0x2000	ROM	Read only memory.

Table 1: Memory Map

3 Testing Schematic

The schematic for testing the Z23 processor can be seen in Figure 2. The wiring for the GPIO pins is based of the program in `src/test.S` which blinks the an LED using the GPIO0 pin. Note that IO pins 1 through 4 are skipped as they are used for the project select feature of the NEBULA chip. IO pins 0 and 5-19 are used for the 16 address bits. IO pins 20-27 are used for the 8 data bits. IO pin 29 is the write/!read pin which is used to control whether the CPU wants to write or read at the address. IO pin 30 is the dedicated interrupt pin. IO pins 30-37 are user programmable GPIO pins. Note that the schematic doesn't reflect the pin out of the Raspberry Pi Pico which is has a few ground pins which aren't shown in the schematic and should be skipped. The pinout of the Pico can be seen in Figure 3.

Hardware Bug Discovered During Testing

During testing, it was discovered that instructions which attempt to write to memory seem to halt the CPU. This is not intended behavior, and is likely due to a hardware bug in the memory controller portion of the design. Instructions which were tested to have failed are `push de`, `push bc`, and `ld (0x3000), a`. Programs which do not write to memory should work fine. If you are able to get memory writes to work properly, please contact me at singh956@purdue.edu. Thank you.

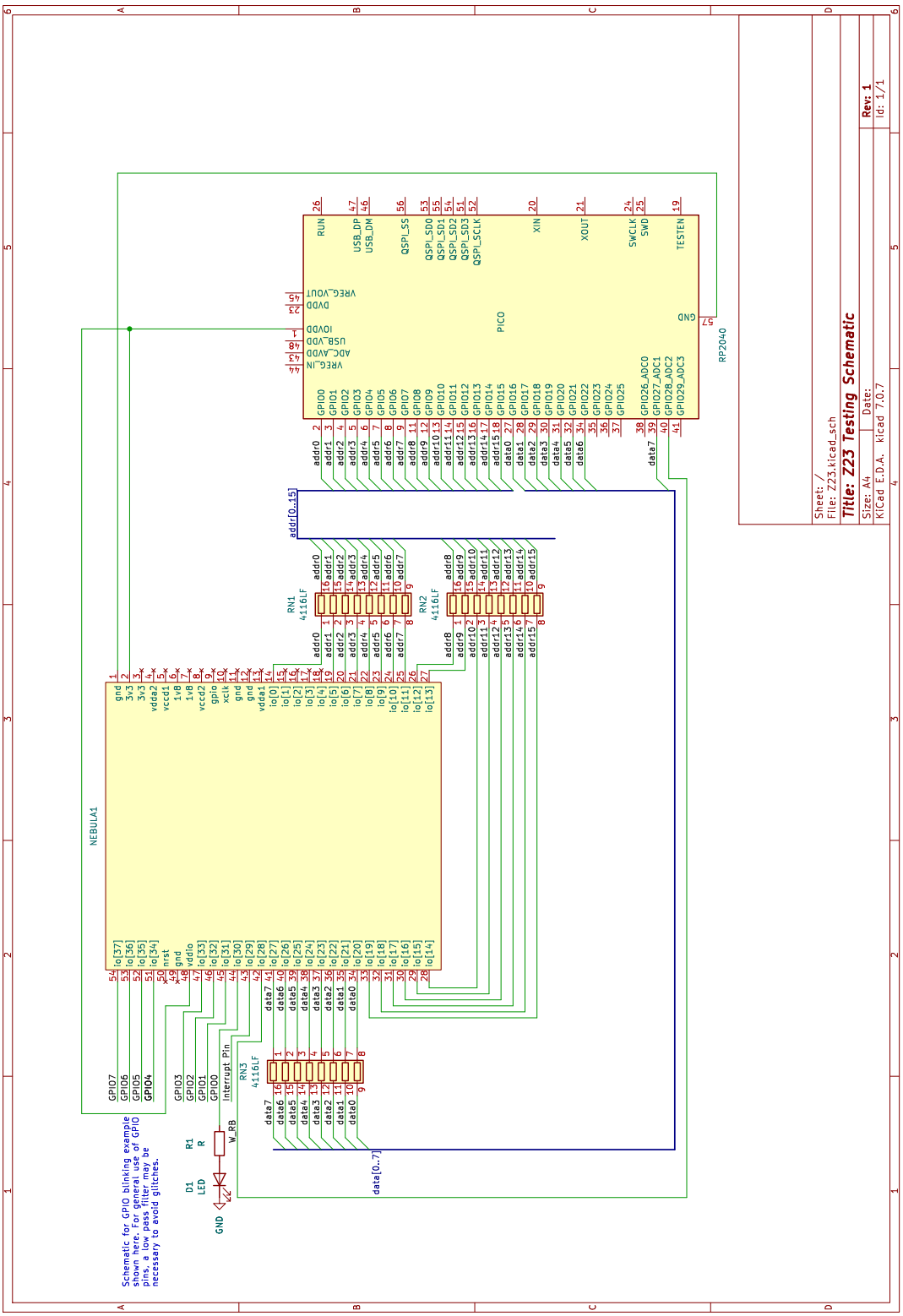


Figure 2: Testing Schematic

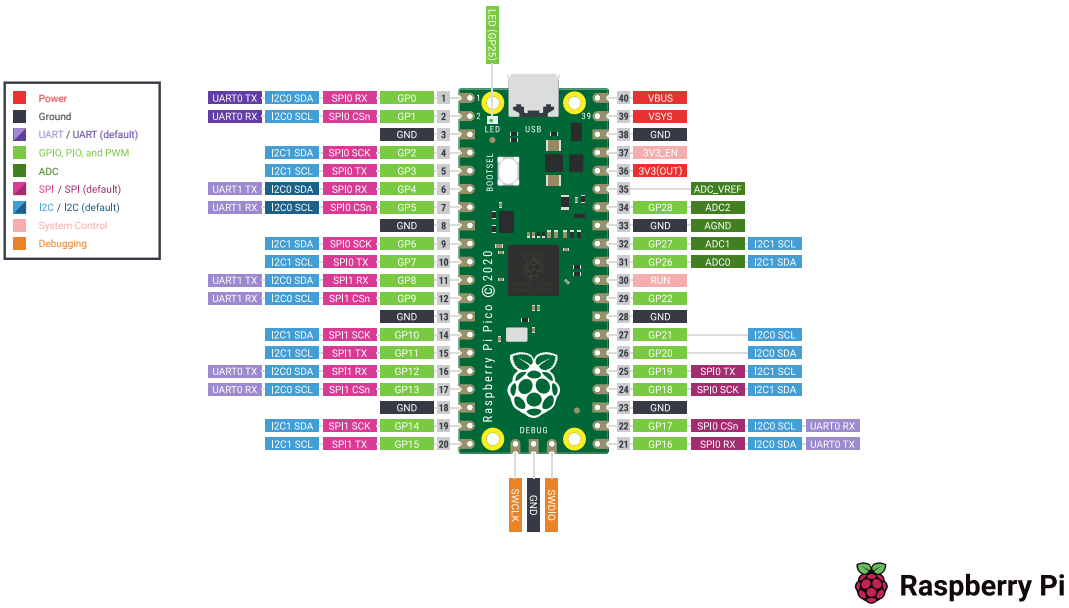


Figure 3: Raspberry Pi Pico Pinout

4 Building and Flashing the Pico

All code related to testing the Z23 can be found at this Github repository. While you are building and flashing the Pico, make sure to keep the Z23 chip reset by shorting the nRST pin to ground. In order to build the project, you must have cmake, the Pico SDK, and the ARM GNU Toolchain installed. Additionally, the `$PICO_SDK_PATH` must point to the `lib/` folder in your SDK installation. The steps to build the program for the Pico are shown below.

1. `git submodule update --init --recursive`
2. `pushd zasm && make zasm && popd`
3. `mkdir build && pushd build && cmake .. && make && popd`

Once you run these commands, a .uf2 file will appear in the `build/` directory. Hold the bootsel button on the Pico as you plug it into your computer, and it should appear as a media drive in your file explorer. Drag the .uf2 into the drive to flash the Pico. Unplug and replug the Pico from your computer to ensure it properly starts up.

5 Testing Files

5.1 src/test.S

This is the default test case which will be built by default. It blinks the GPIO0 pin repeatedly by toggling the GPIO0 control register and then waiting in spin loops.

5.2 src/blink_gpio.S

This is very similar to `src/test.S`, but it has a smaller toggle delay which can be seen easier using an AD2.

5.3 src/spi_oled.S

This file implements a simple SPI interface using GPIO0 as nCS, GPIO1 as SDO, and GPIO2 as SCL which is designed to be used with the 1602A OLED display which is used in ECE 36200. It was developed before the memory write bug was discovered, however, in theory it should work and display the letter 'a' on the screen.

5.4 src/z23.S

This file is a simple include file for any program which wishes to use the `reti` instruction to return from an interrupt. The Z23 has a nonstandard decoding of this instruction.

5.5 src/main.c

This file is the main file which is flashed onto the Pico. It essentially sits in a loop and responds to the bus requests from the Z23 as they come in. Note that it relies on overclocking the Pico to achieve acceptable response speeds. If you are uncomfortable with this, you can comment out line 100 which calls `set_sys_clock_khz`. After the Pico has been successfully flashed, the LED near the bootsel button should light up.

5.6 src/rom.h

This file is generated by running the `zasm` assembler on `src/test.S` before building the rest of `src/main.c`. It defined an array of bytes which correspond to the ROM data generated by the assembler.

5.7 layout/Z23

The KiCad schematic files for the Z23 testing schematic.