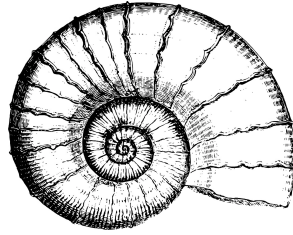


# The Shell CPU

## Memory Modules and Final Testing



Jonathan Rice Shelley II  
November 18, 2020

Version 1.0

## Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>Test Program</b>	<b>2</b>

## 1 Overview

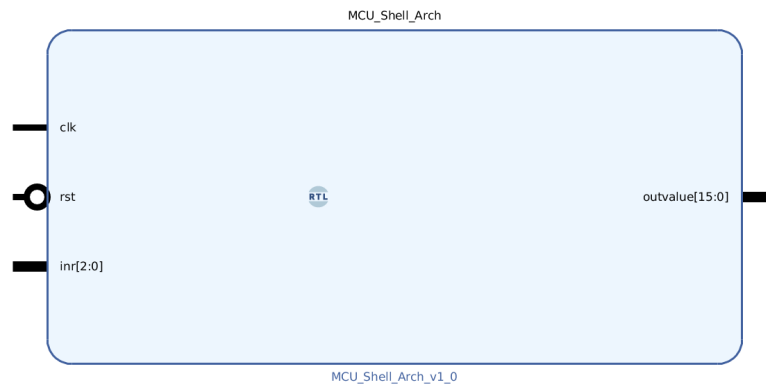


Figure 1: MCU Shell Architecture.

This document details the final implementation of the Shell CPU into a MCU top level module (Figure 1). The Design has three inputs and one 16-bit output port. Embedded program memory and data memory have been added to the design using block RAMs. The three bit inr line is used to select which of the eight internal CPU registers is displayed on the 16-bit bus outvalue. The VHDL for each module is too long to be listed and can be found with other documentation here.

## 2 Test Program

```

1 section .data
2     sp    dw 0x00FF
3     num1  dw 6
4     num2  dw 7
5     ans   dw 0
6
7 section .text
8     ; set stack pointer
9     movi  r1, sp
10    lw    r1, r1
11    lsp   r1
12
13    zero  r1
14    zero  r2
15
16    ; Load values to multiply from memory
17    movi  r1, num1
18    lw    r1, r1
19
20    movi  r2, num2
21    lw    r2, r2
22
23    ; put params on stack
24    push  r1
25    push  r2
26
27    ; jump to mul sub routine
28    zero  r4
29    addi  r4, mul
30    lw    r4, r4
31    jalr  r5, r4
32
33    ; pop multiplication result off stack
34    pop   r1
35
36    ; store multiplication result in memory
37    movi  r2, ans
38    sw    r1, r2
39
40    ; exit program
41    hlt
42
43    ; @params two 16-bit numbers on stack
44    ; @return 16-bit result on stack
45    mul:
46    pop   r1
47    pop   r2
48    zero  r3
49    add   r3, r3, r1
50    zero  r1
51    _mul_loop:
52    cmp   r0, r2
53    beq   exit
54    add   r1, r1, r3
55    subi  r2, 1
56    cmp   r0, r0
57    beq   _mul_loop
58    exit:
59    push  r1
60    jalr  r0, r5
  
```

```

1 lui 1 0
2 addi 1 0
3 lw 1 1
4 lsp 1
5 xor 1 1 1
6 xor 2 2 2
7 lui 1 0
8 addi 1 1
9 lw 1 1
10 lui 2 0
11 addi 2 2
12 lw 2 2
13 push 1
14 push 2
15 xor 4 4 4
16 addi 4 mul
17 lw 4 4
18 jalr 5 4
19 pop 1
20 lui 2 0
21 addi 2 3
22 sw 1 2
23 hlt
24 mul:
25 pop 1
26 pop 2
27 xor 3 3 3
28 add 3 3 1
29 xor 1 1 1
30 _mul_loop:
31 cmp 0 2
32 beq exit
33 add 1 1 3
34 subi 2 1
35 cmp 0 0
36 beq _mul_loop
37 exit:
38 push 1
39 jalr 0 5
  
```

```

1 0110000001000000
2 1101000001000000
3 1010000001001000
4 1110010001000000
5 1100100001001001
6 1100100001001010
7 0110000001000000
8 1101000001000001
9 1010000001001000
10 0110000001000000
11 11010000010000010
12 10100000010010000
13 1110000001000000
14 1110000001000000
15 1100100100100100
16 1101000100000100
17 1010000100100000
18 0100000101100000
19 1110001001000000
20 0110000001000000
21 11010000010000011
22 1000000001010000
23 0000000000000000
24 1110001001000000
25 1110001010000000
26 1100100011011011
27 1100000011011001
28 1100100001001001
29 1101010000010000
30 001000000000101
31 1100000001001011
32 1101001010000001
33 1101010000000000
34 00100111111011
35 1110000001000000
36 0100000001010000
37 0000000000000000
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
  
```

Figure 2: Test Program

A test program is written to demonstrate the functionality of system memory (Figure 2). The test program computes seven times six. The values of seven and six are loaded from memory multiplied and the result is written back to memory. The stack is used to pass data to and from the multiplication subroutine. The stack pointer is set to 255 at the beginning of the program. This test program adequately test all CPU memory functionality.

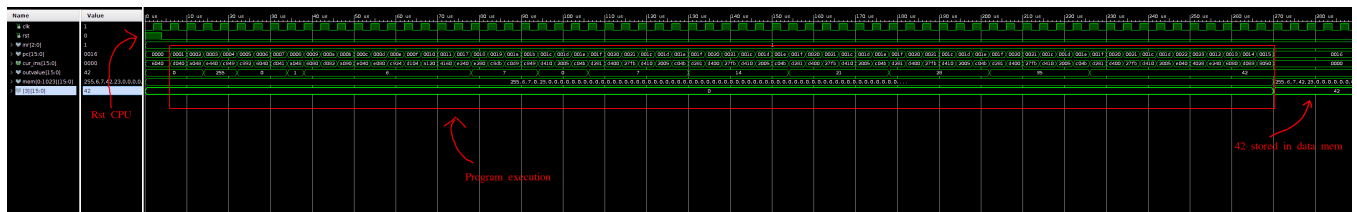


Figure 3: Test Program Simulation

Address	Disasm	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	4
---------	--------	---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	---

Figure 4: Test Program Simulation With Control Signals