CS0 2130

Overview

Daniel G. Graham PhD

1. Recursive Program
2. Overview from System to Gates

RECURSIVE EXAMPLE

Let's write a recursive function that sum all the positive numbers up to including the number x

3+ 2 + 1

UNIVERSITY of VIRGINIA | ENGINEERING

RECURSIVE EXAMPLE

Let's write a recursive function that sum all the positive numbers up to including the number x

3+ 2 + 1

```
int sum(int x)
        if x == 0:
            return 0
        return x+ sum(x-1)
```


UNIVERSITY of VIRGINIA | ENGINEERING

WHAT ABOUT THE FUNCTION PARAMETERS

We need to define a calling convention. The rules that we'll follow when we call a function.

1. For our simple functions are limited to 2 parameters.
2. The first parameter will be stored in R2
3. The second parameter will be stored in R3
4. The return value of the function will be stored in R0
5. If the function uses any other registers save them before modifying them and restore them before returning.

```
input = 0xFF
shiftAmount = 0x02
output = left_shift(input, shiftAmount)
```

```
R2 = 0xFF
R3 = 0x02
call left_shift
R0 //Contains result
```

RECURSIVE EXAMPLE

Let's write a recursive function that sum all the positive numbers up to including the number x

3+ 2 + 1

```
int sum(int x)
        if x == 0:
            return 0
        return x+ sum(x-1)
```

```
R2 = 03
CALL  SUM
R1 =  Memory of RET
If R2 <= 0 PC =  R1
R0 += R2
R2+= -1
CALL 20
RET
```

The first parameter will be stored in R2

```
68 03
82 XX  //ex 20
64 XX //ex 28
79
12
69 FF
82 20
83
```


UNIVERSITY of VIRGINIA | ENGINEERING

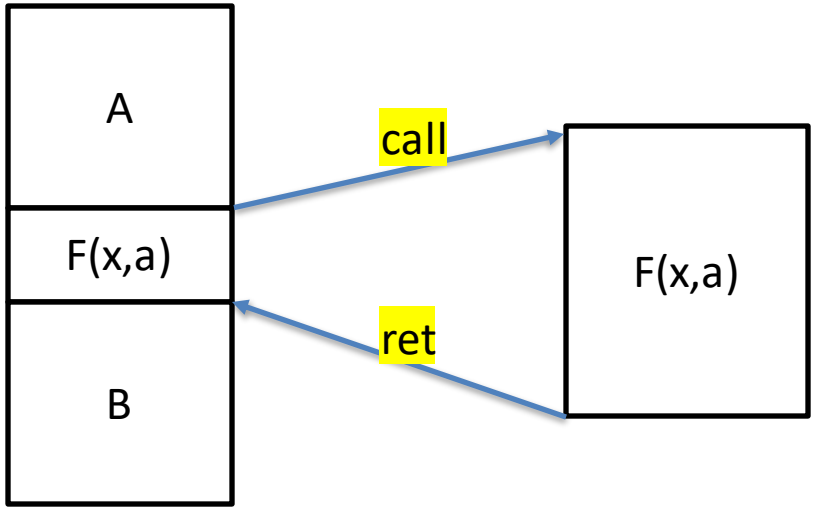
DEFINING A NEW INSTRUCTION

Let's create a new instruction that will both save the location to return and jump to the beginning of the function. We'll name this our **call** instruction

Save pc+2 , set pc = M[pc+1]

Let's also create an instruction that sets the PC back to the saved. We'll name this our return instruction or ret for short

pc = Saved Value



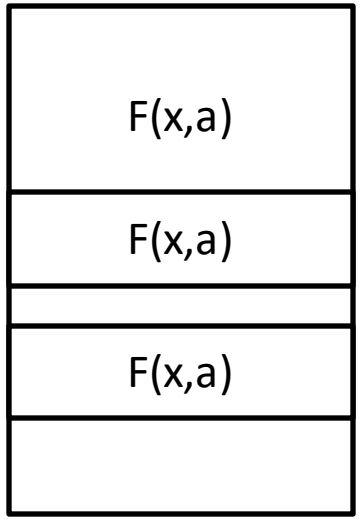
UNIVERSITY of VIRGINIA | ENGINEERING

ISA EXTENDED BY SETTING R BIT TO 1

Reserve	icode	b	operation
1	0		
		0	---- Coming Soon---
		1	----Coming Soon ----
		2	Place pc+2 onto the [Memory], set pc = M[pc+1]
		3	pc = a value from [Memory]

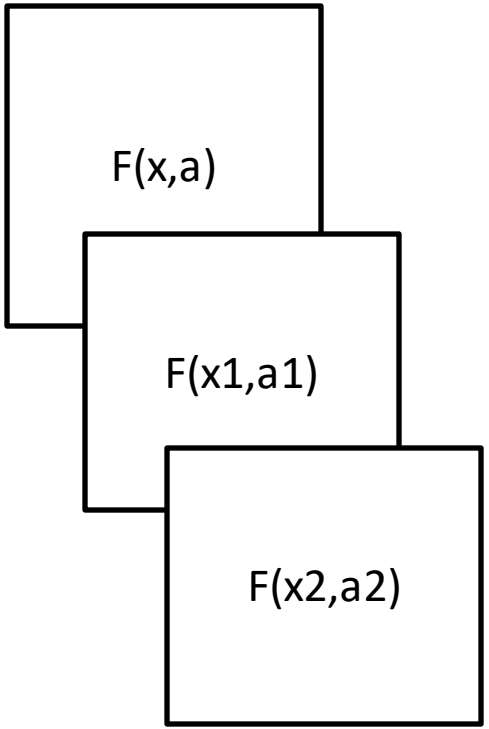
More details on Memory coming soon

WHAT ABOUT RECURSIVE FUNCTIONS



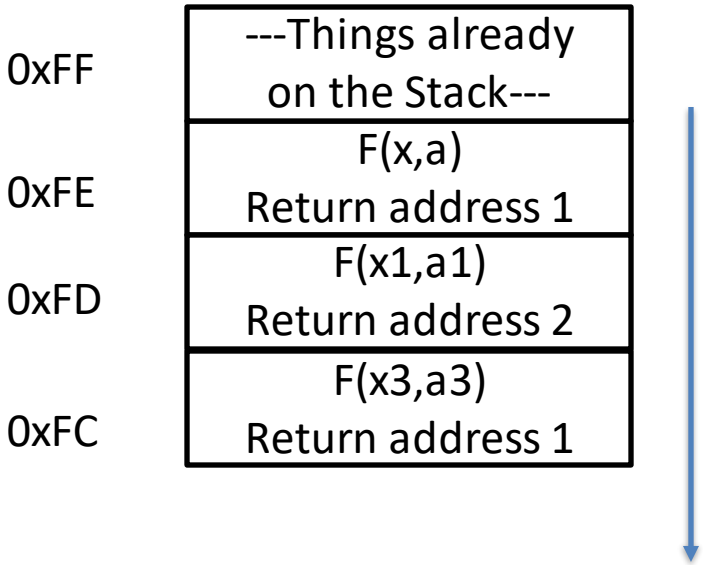
What about recursive functions? Functions that call themselves

Now we need to keep track of both the location return to (multiple function calls and the register state of function before the call)



THE STACK

We are going to a region of memory that will hold the stack of function states and their associated return addresses.

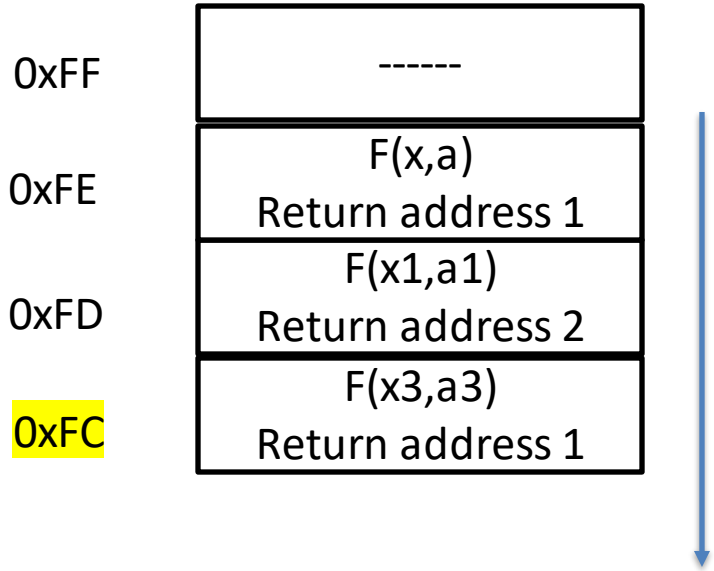


By convention keep adding new things to the stack by growing it to lower addresses

THE STACK

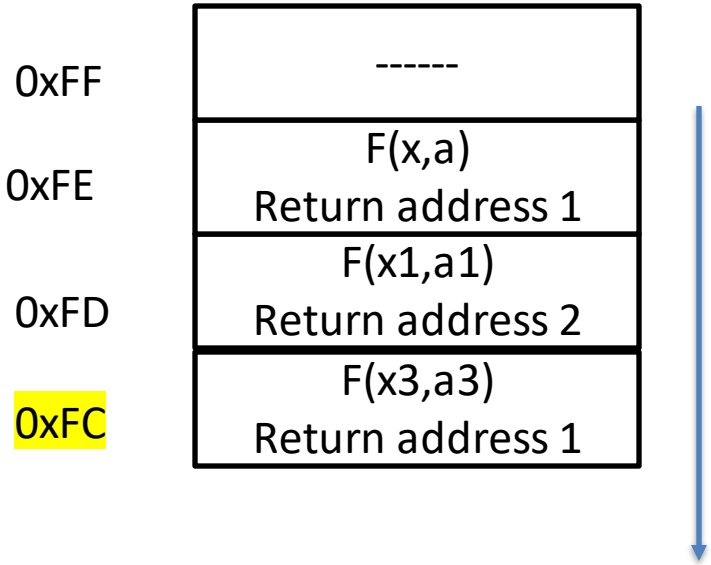
RSP | 0xFC

We also define a new register that holds the location of the TOP of the stack in memory. We'll name this register RSP



PUSH AND POP INSTRUCTIONS

RSP | 0xFC



We'll also create two instructions that will add and remove values from the stack.

The push instruction will decrement the RSP and to the top of the stack

Example push(0x04)

UNIVERSITY of VIRGINIA | ENGINEERING

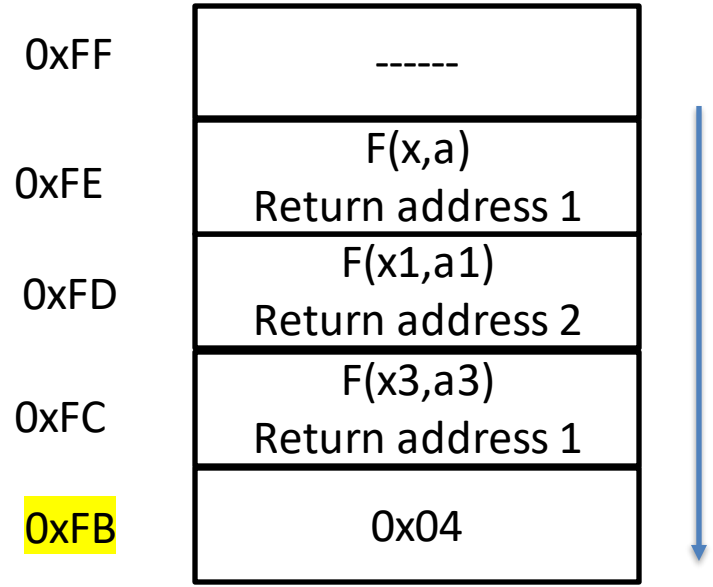
PUSH AND POP INSTRUCTIONS

RSP | 0xFB

We'll also create two instructions that will add and remove values from the stack.

The push instruction will decrement the RSP and to the top of the stack

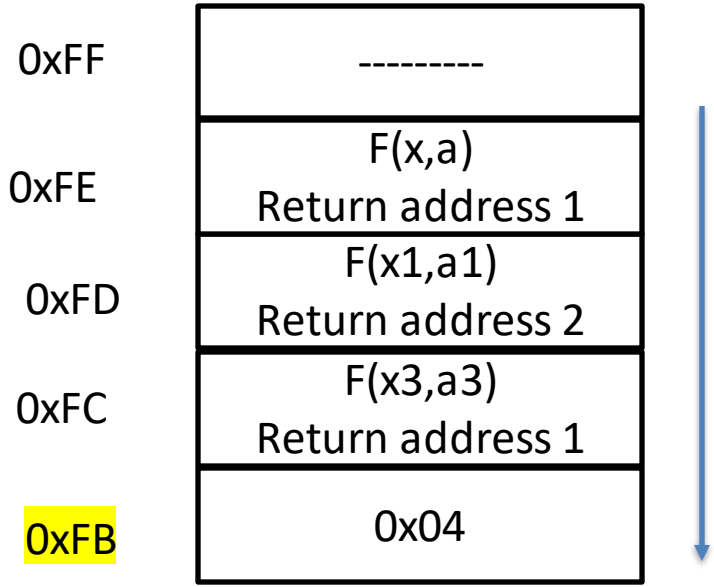
Example push(0x04)



UNIVERSITY of VIRGINIA | ENGINEERING

PUSH AND POP INSTRUCTIONS

RSP | 0xFB



We'll also create two instructions that will add and remove values from the stack.

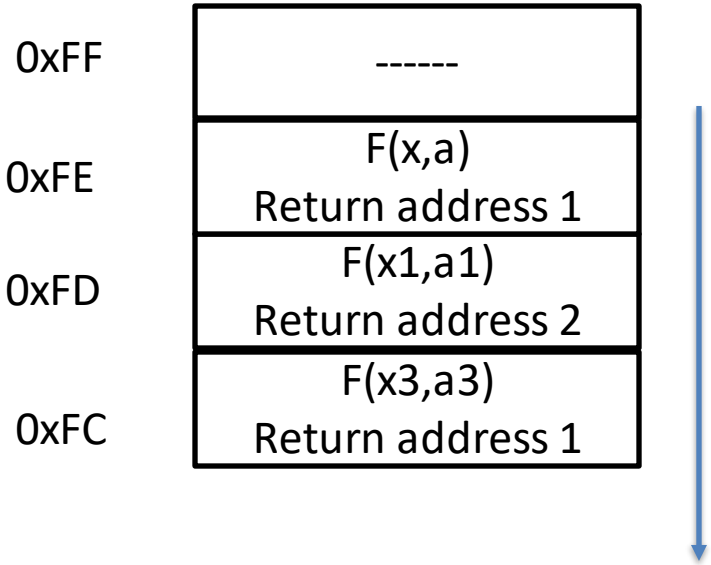
While the pop instruction increments RSP and returns the value at the top of the stack

Example x = pop()

UNIVERSITY of VIRGINIA | ENGINEERING

PUSH AND POP INSTRUCTIONS

RSP | 0xFC



We'll also create two instructions that will add and remove values from the stack.

While the pop instruction returns the value at the top of the stack and **then** increments RSP

Example x = pop() returns 0x04

UNIVERSITY of VIRGINIA | ENGINEERING

WHAT ABOUT THE FUNCTION PARAMETERS

We need to define a calling convention. The rules that we'll follow when we call a function.

1. For our simple processor functions are limited to 2 parameters.
2. The first parameter will be stored in R2
3. The second parameter will be stored in R3
4. The return value of the function will be stored in R0
5. If the function uses any other registers save them before modifying them and restore them before returning.

```
input = 0xFF
shiftAmount = 0x02
output = left_shift(input, shiftAmount)
```

```
R2 = 0xFF
R3 = 0x02
call left_shift
R0 //Contains result
```

ISA EXTENDED BY SETTING R BIT TO 1

icode	b	operation
0		
	0	Decrement rsp and push the contents of rA to the stack
	1	Pop the top value from the stack into rA and increment rsp
	2	Push pc+2 onto the stack, set pc = M[pc+1]
	3	pc = pop the top value from the stack

If b is not 2 or 3 update the pc as normal. A is not used either (Let's do some encoding examples)

UNIVERSITY of VIRGINIA | ENGINEERING

RECURSIVE EXAMPLE

Let's write a recursive function that sum all the positive numbers up to including the number x

3+ 2 + 1

```
int sum(int x)
        if x == 0:
            return 0
        return x+ sum(x-1)
```

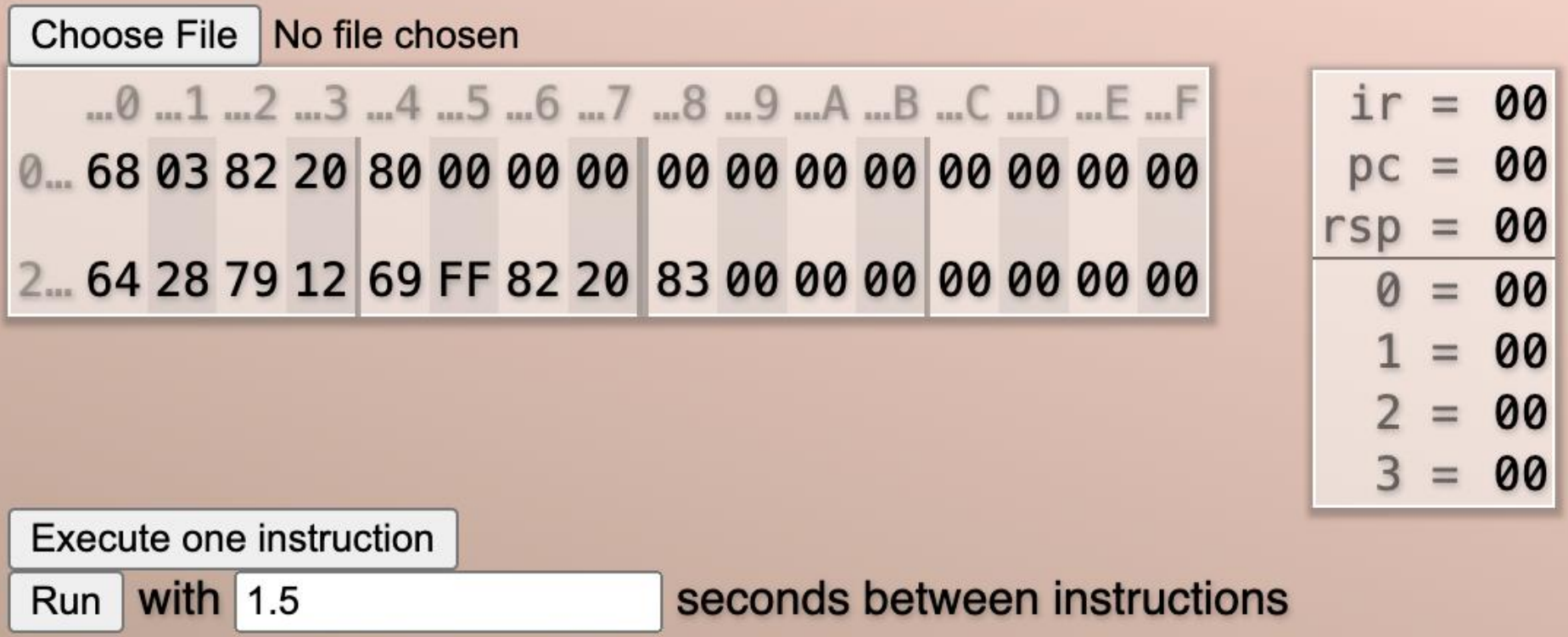
```
R2 = 03
CALL  SUM
R1 =  Memory of RET
If R2 <= 0 PC =  R1
R0 += R2
R2+= -1
CALL 20
RET
```

The first parameter will be stored in R2

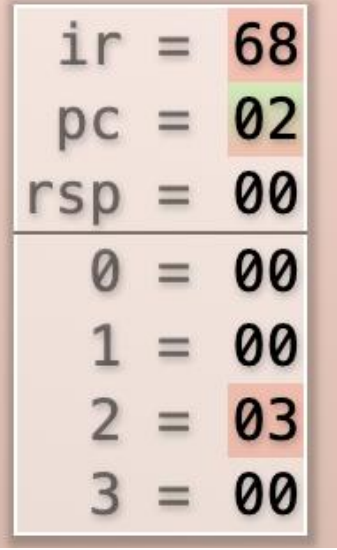
```
68 03
82 XX  //ex 20
64 XX //ex 28
79
12
69 FF
82 20
83
```


UNIVERSITY of VIRGINIA | ENGINEERING

LET'S STEP THROUGH THIS EXECUTION



	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00



ir =	68
pc =	02
rsp =	00
0 =	00
1 =	00
2 =	03
3 =	00

R2 = 03

CALL SUM

R1 = Memory of RET

If R2 <= 0 PC = R1

R0 += R2

R2+= -1

CALL 20

RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	04

ir =	82
pc =	20
rsp =	FF
0 =	00
1 =	00
2 =	03
3 =	00

R2 = 03

CALL SUM

R1 = Memory of RET

If R2 <= 0 PC = R1

R0 += R2

R2+= -1

CALL 20

RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	04

ir	=	64
pc	=	22
rsp	=	FF
0	=	00
1	=	28
2	=	03
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	04

ir	79
pc	23
rsp	FF
0	00
1	28
2	03
3	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	04

ir =	12
pc =	24
rsp =	FF
0 =	03
1 =	28
2 =	03
3 =	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	04

ir	=	69
pc	=	26
rsp	=	FF
0	=	03
1	=	28
2	=	02
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	28	04

ir	=	82
pc	=	20
rsp	=	FE
0	=	03
1	=	28
2	=	02
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	28	04

ir =	64
pc =	22
rsp =	FE
0 =	03
1 =	28
2 =	02
3 =	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	28	04

ir	=	79
pc	=	23
rsp	=	FE
0	=	03
1	=	28
2	=	02
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	28	04

ir	=	12
pc	=	24
rsp	=	FE
0	=	05
1	=	28
2	=	02
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	00	28	04

ir	=	69
pc	=	26
rsp	=	FE
0	=	05
1	=	28
2	=	01
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	28	28	04

ir =	82
pc =	20
rsp =	FD
0 =	05
1 =	28
2 =	01
3 =	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	28	28	04

ir =	64
pc =	22
rsp =	FD
0 =	05
1 =	28
2 =	01
3 =	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	28	28	04

ir	=	79
pc	=	23
rsp	=	FD
0	=	05
1	=	28
2	=	01
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	28	28	04

ir =	12
pc =	24
rsp =	FD
0 =	06
1 =	28
2 =	01
3 =	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	00	28	28	04

ir =	69
pc =	26
rsp =	FD
0 =	06
1 =	28
2 =	00
3 =	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	28	28	28	04

ir =	82
pc =	20
rsp =	FC
0 =	06
1 =	28
2 =	00
3 =	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	28	28	28	04

ir	=	64
pc	=	22
rsp	=	FC
0	=	06
1	=	28
2	=	00
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	28	28	28	04

ir	=	79
pc	=	28
rsp	=	FC
0	=	06
1	=	28
2	=	00
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

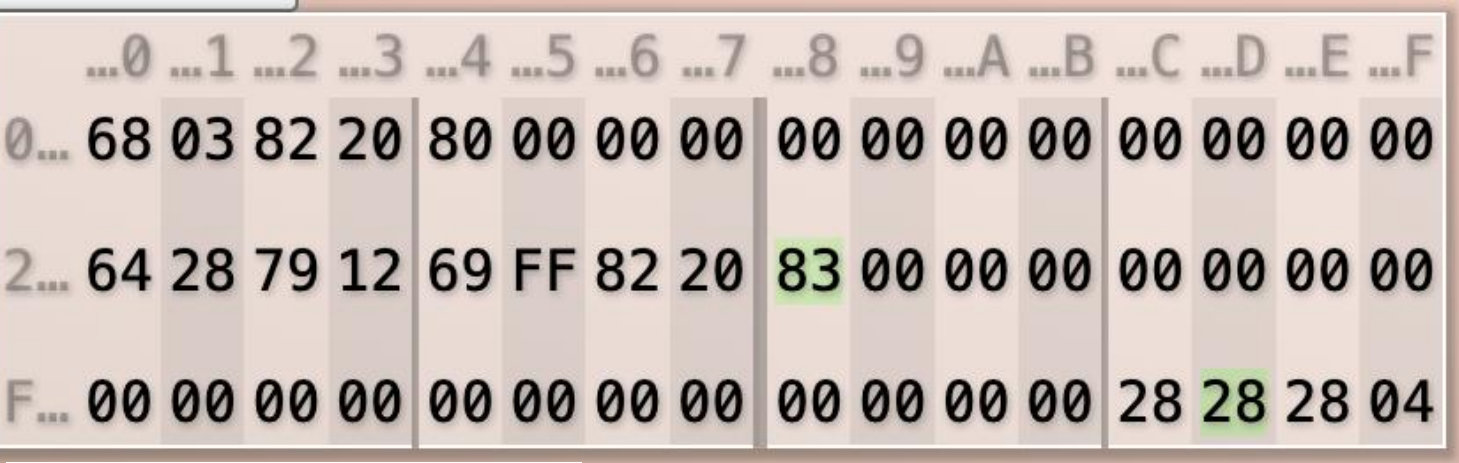
UNIVERSITY of VIRGINIA | ENGINEERING

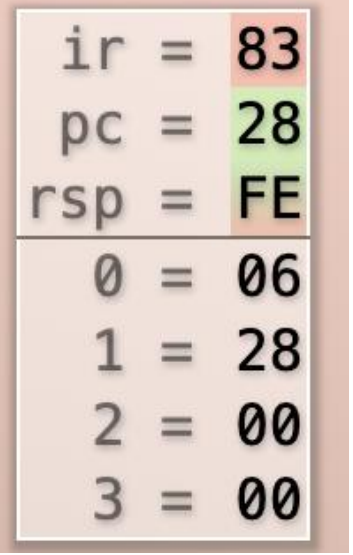
	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	28	28	28	04

ir	=	83
pc	=	28
rsp	=	FD
0	=	06
1	=	28
2	=	00
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING





R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	28	28	28	04

ir =	83
pc =	28
rsp =	FF
0 =	06
1 =	28
2 =	00
3 =	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

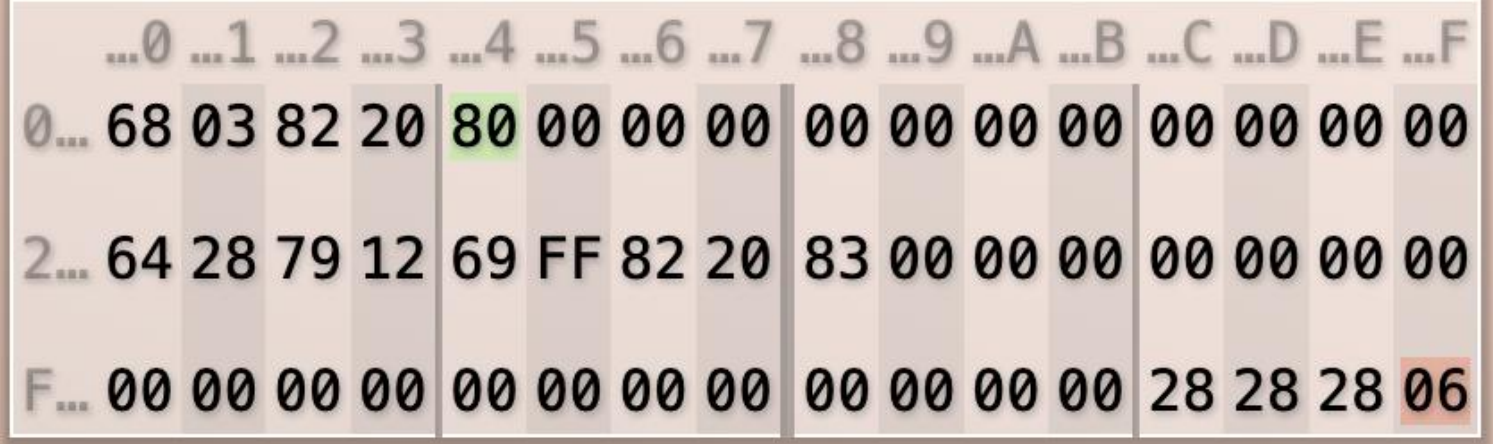
UNIVERSITY of VIRGINIA | ENGINEERING

	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	28	28	28	04

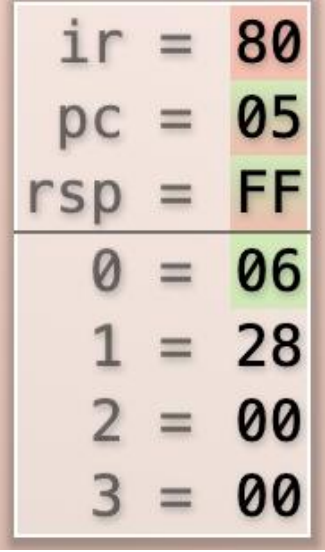
ir =	83
pc =	04
rsp =	00
0 =	06
1 =	28
2 =	00
3 =	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING



	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	68	03	82	20	80	00	00	00	00	00	00	00	00	00	00	00
2…	64	28	79	12	69	FF	82	20	83	00	00	00	00	00	00	00
F…	00	00	00	00	00	00	00	00	00	00	00	00	28	28	28	06



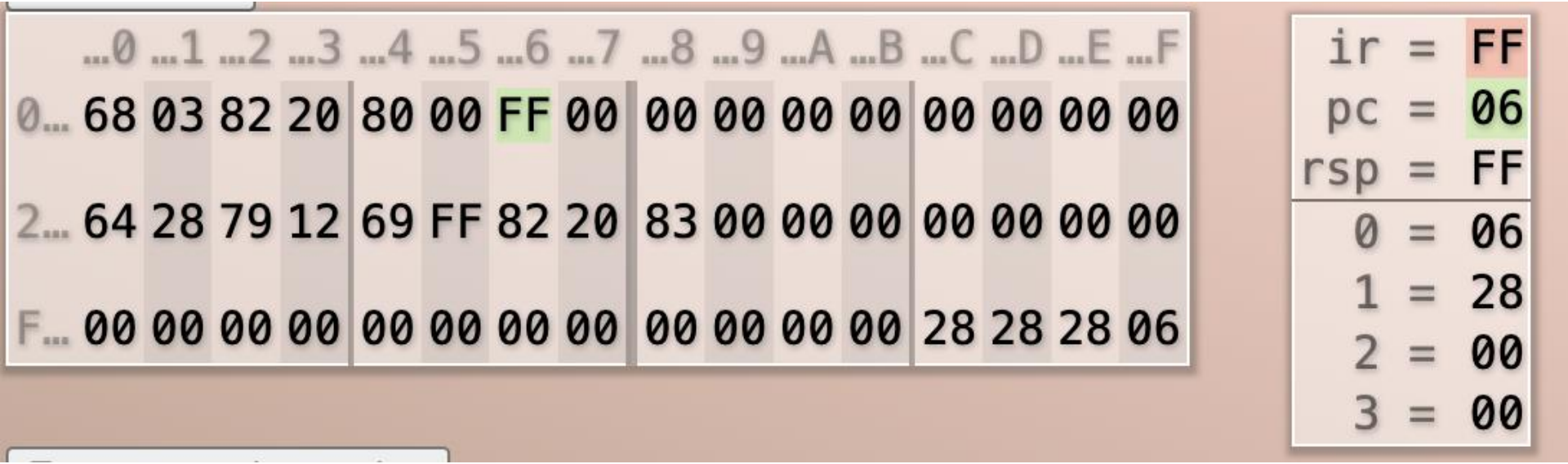
ir	=	80
pc	=	05
rsp	=	FF
0	=	06
1	=	28
2	=	00
3	=	00

R2 = 03
CALL SUM
R1 = Memory of RET
If R2 <= 0 PC = R1
R0 += R2
R2+= -1
CALL 20
RET

UNIVERSITY of VIRGINIA | ENGINEERING

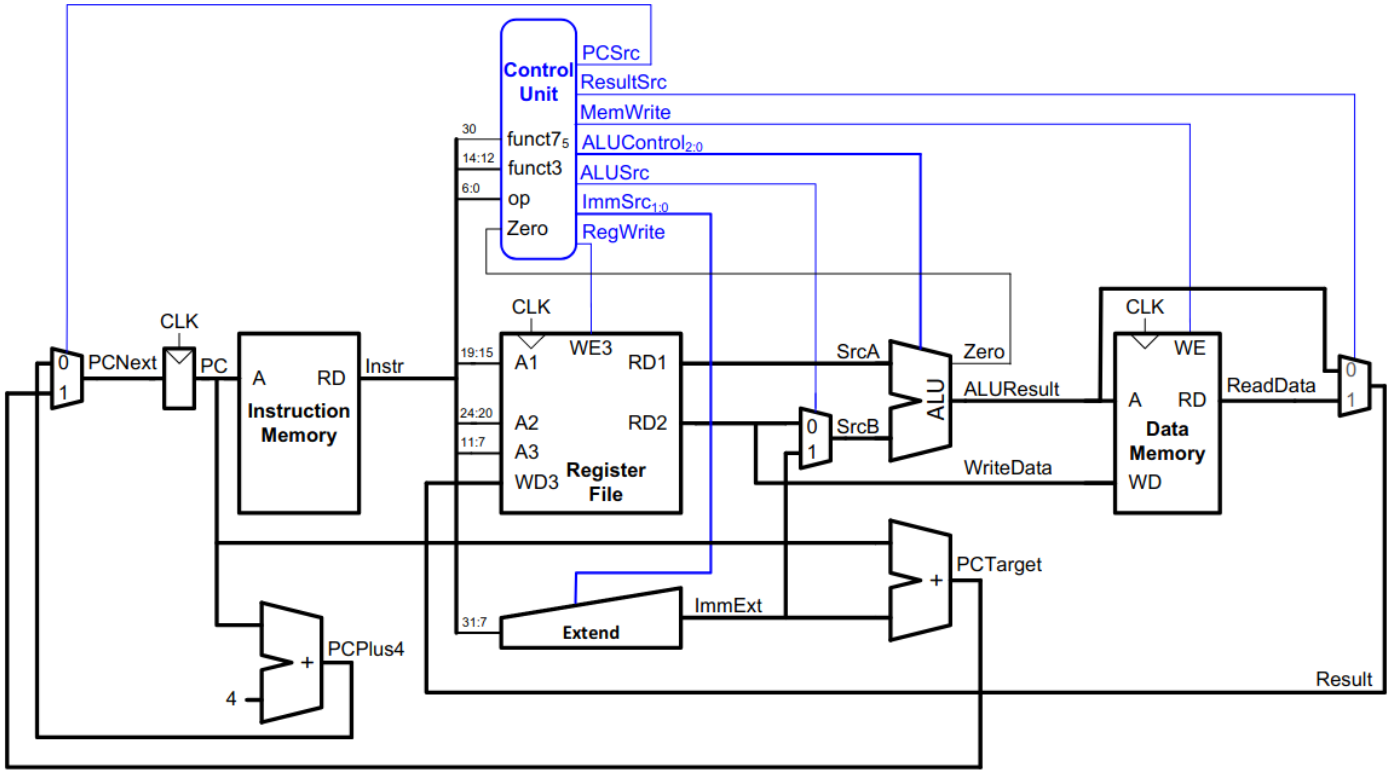
HALT

New Halt instruction

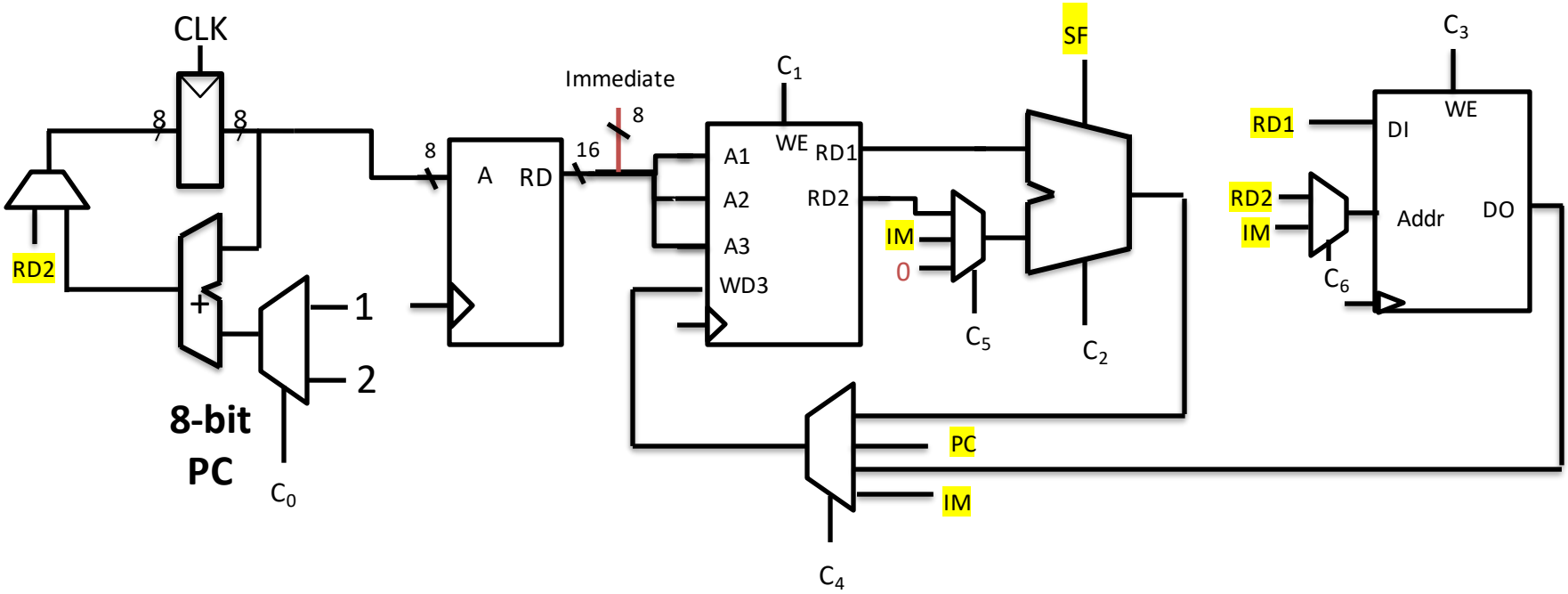


UNIVERSITY of VIRGINIA | ENGINEERING

OUR JOURNEY SO FAR



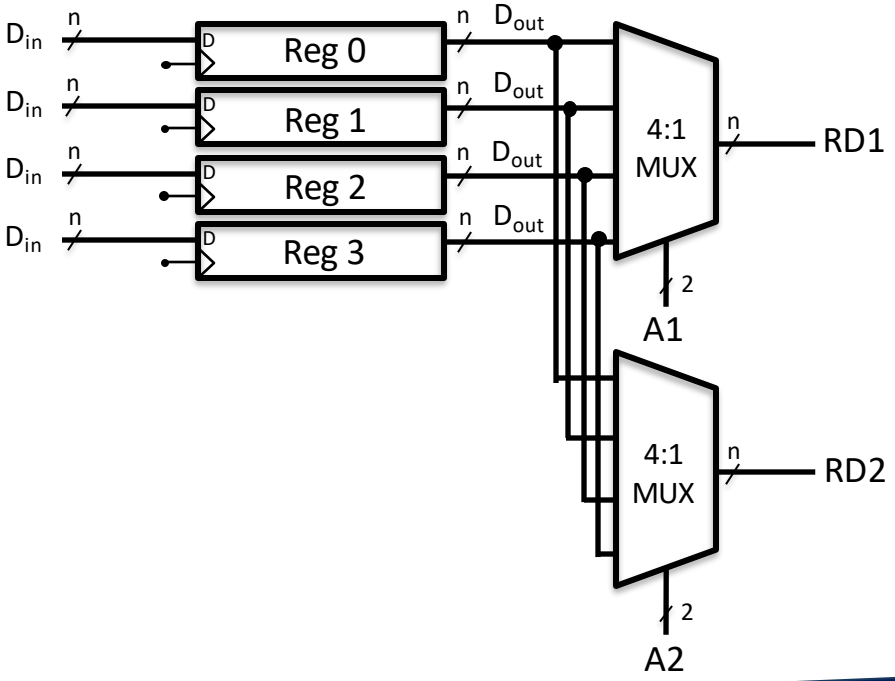
UNIVERSITY of VIRGINIA | ENGINEERING



Write back stages

UNIVERSITY of VIRGINIA | ENGINEERING

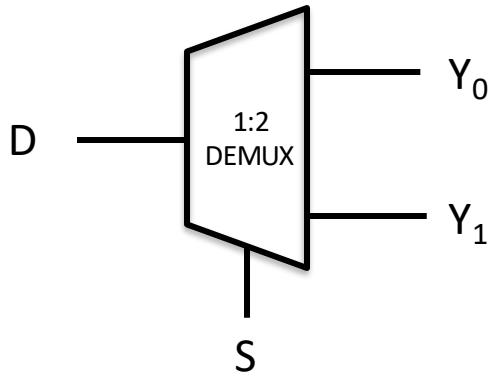
READ FROM A REGISTER FILE



UNIVERSITY of VIRGINIA | ENGINEERING

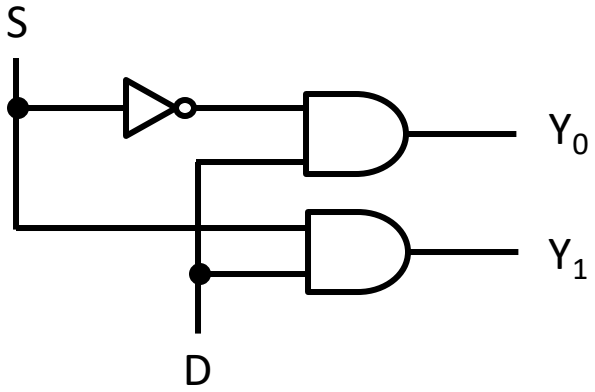
DEMULTIPLEXER (DEMUX)

Example: **1:2 Demux**



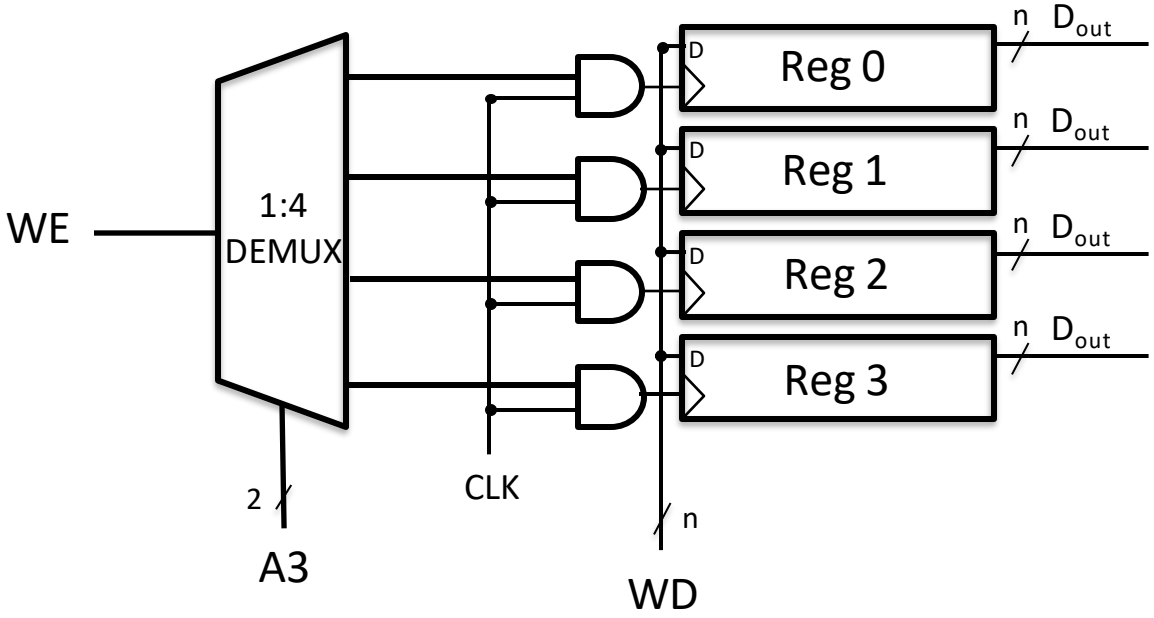
S	Y0	Y1
0	**D**	0
1	0	**D**

- Connects one input to one of the **N** outputs
- **Select** input is **$\log_2 N$ bits** – control input



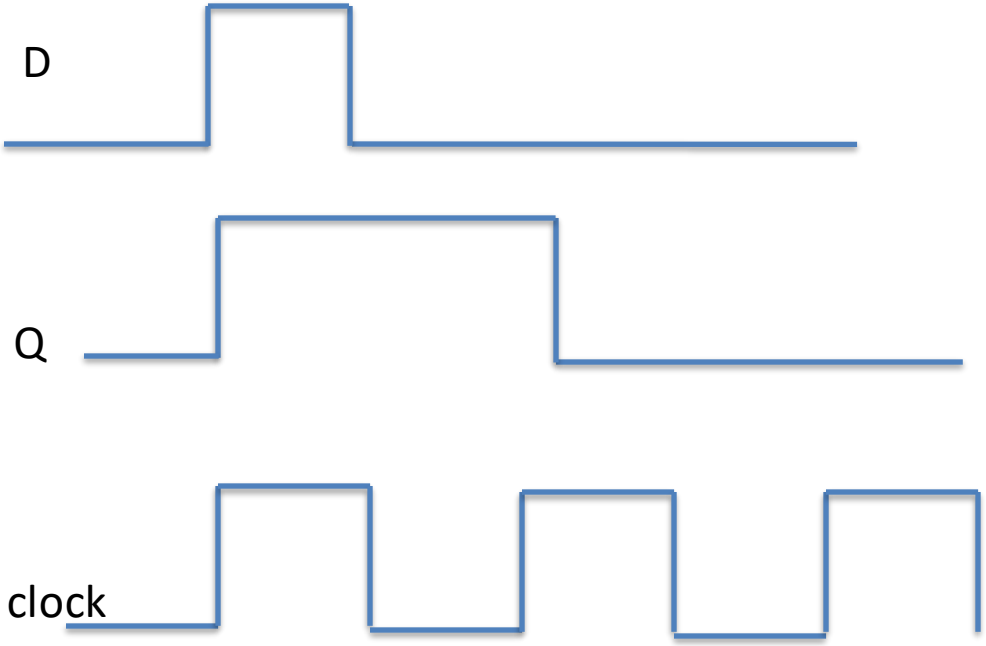
UNIVERSITY of VIRGINIA | ENGINEERING

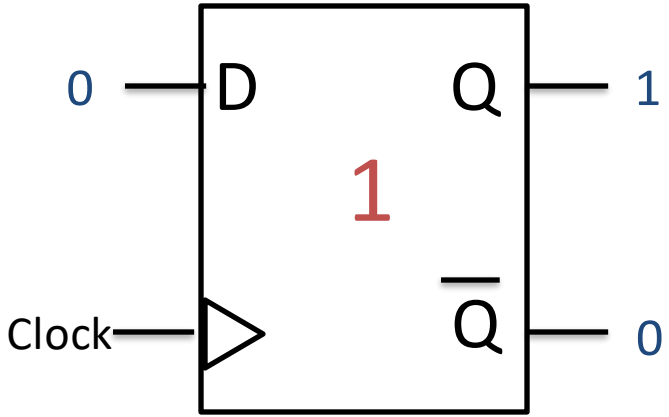
WRITE TO A REGISTER FILE



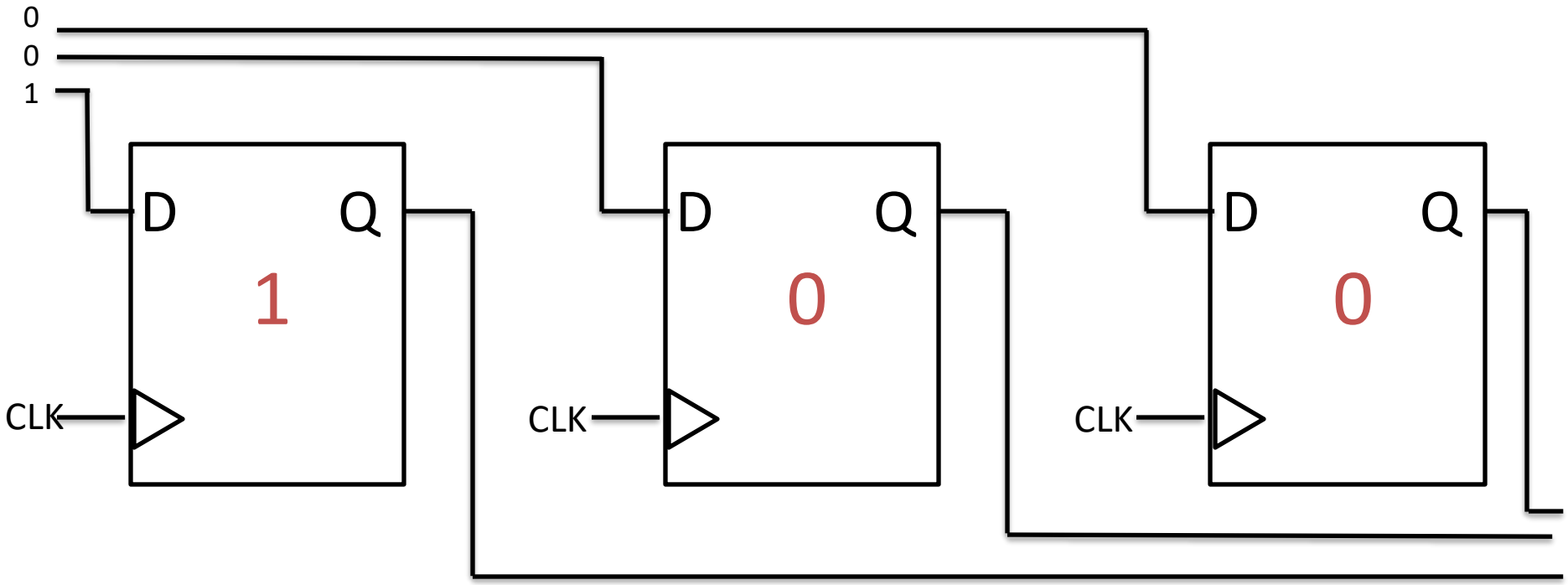
UNIVERSITY of VIRGINIA | ENGINEERING

THE FLIP FLOP HOLD HOLDS THE VALUE FOR A CLOCK CYCLE



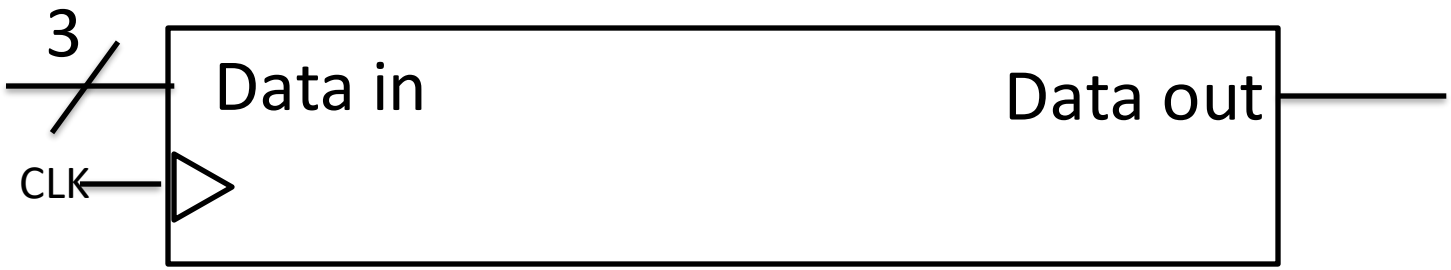


BUILDING A REGISTER FROM FLIP FLOPS

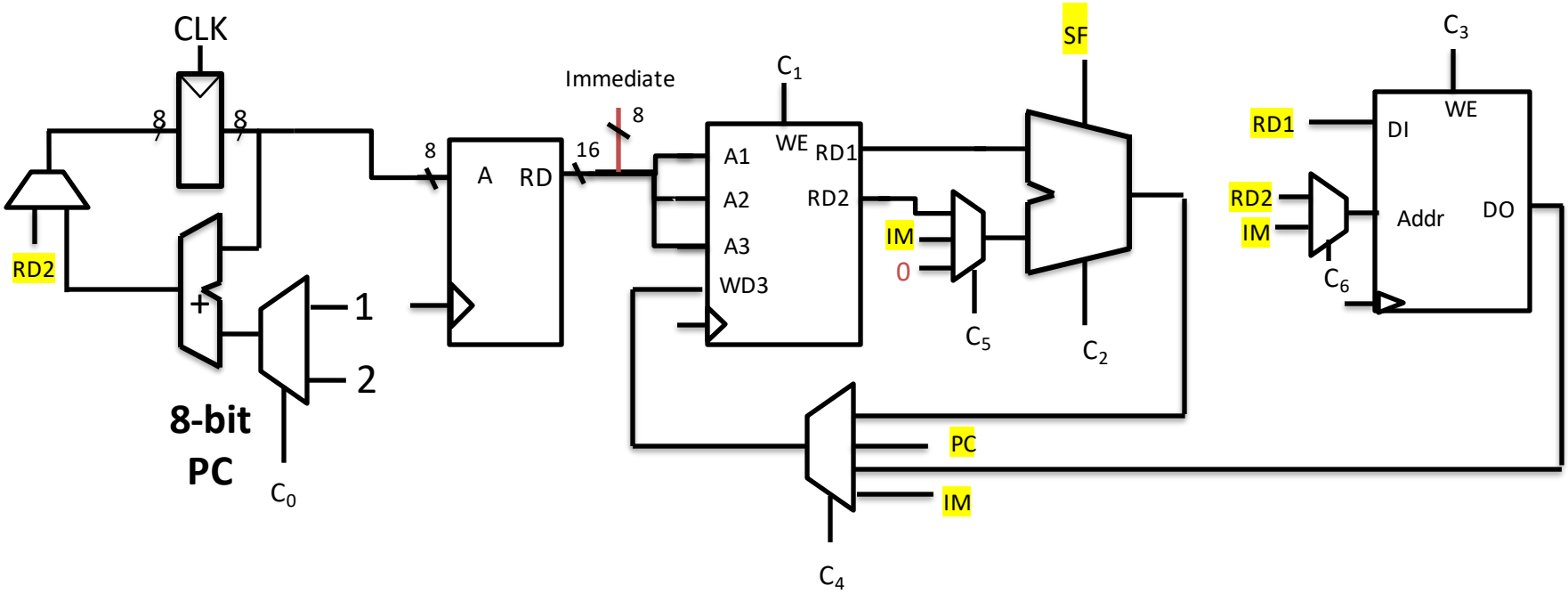


Removed Q (bar) for readability

REGISTER SYMBOLS



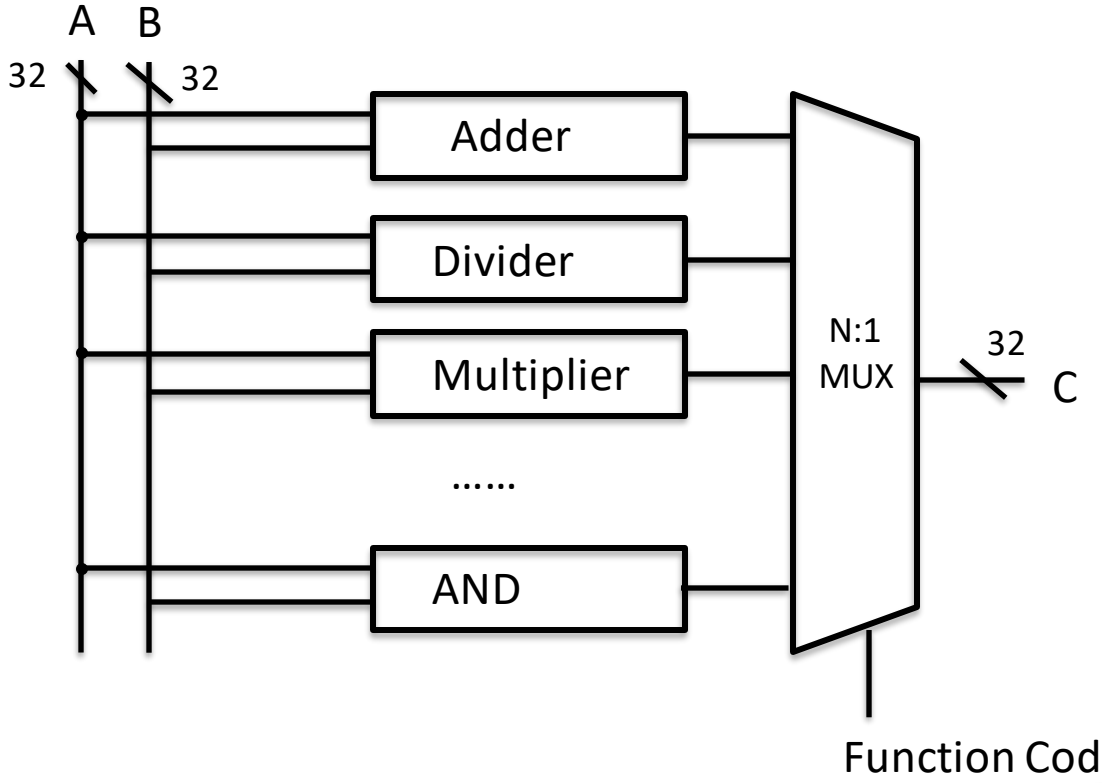
UNIVERSITY of VIRGINIA | ENGINEERING



Write back stages

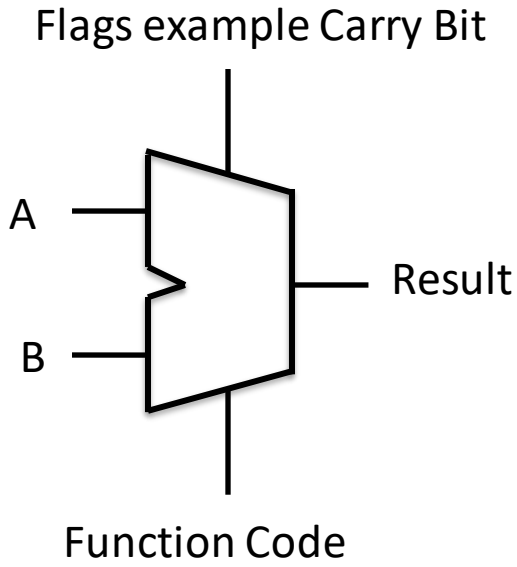
UNIVERSITY of VIRGINIA | ENGINEERING

ARITHMETIC LOGIC UNIT



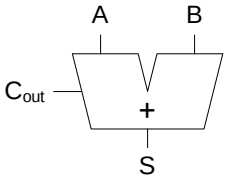
UNIVERSITY of VIRGINIA | ENGINEERING

ALU SYMBOL AND INPUTS



UNIVERSITY of VIRGINIA | ENGINEERING

Half Adder

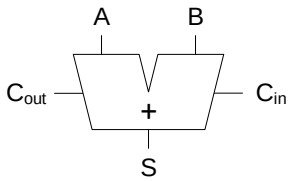


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A \oplus B$

$C_{out} = AB$

Full Adder



C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$S = A \oplus B \oplus C_{in}$

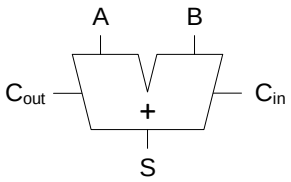
$C_{out} = AB + AC_{in} + BC_{in}$

```
1 1 1 1   ← Carries
  0 1 1 1
+ 1 0 1 1
  0 0 1 0
```

Note on special case 3 input xor.
Draw the three gates. Really
Three xors stacked.

UNIVERSITY of VIRGINIA | ENGINEERING

Full Adder

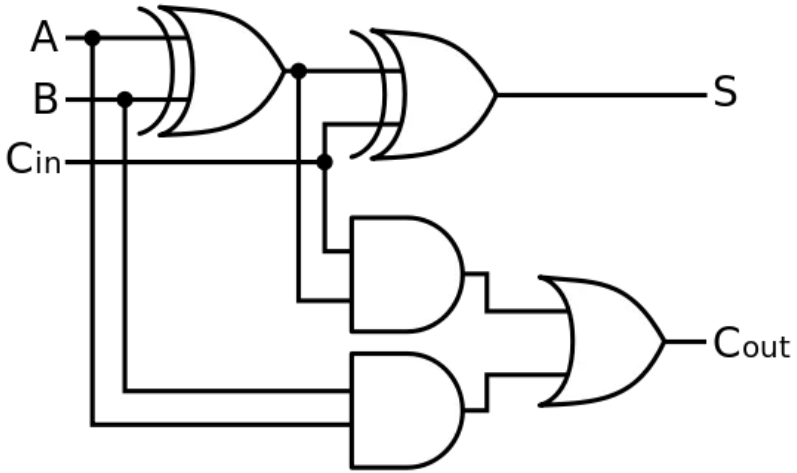


C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$S = A \oplus B \oplus C_{in}$

$C_{out} = AB + AC_{in} + BC_{in}$

```
 1 1 1 1   ← Carries
   0 1 1 1
+  1 0 1 1
 ---------
   1 1 1 0
```

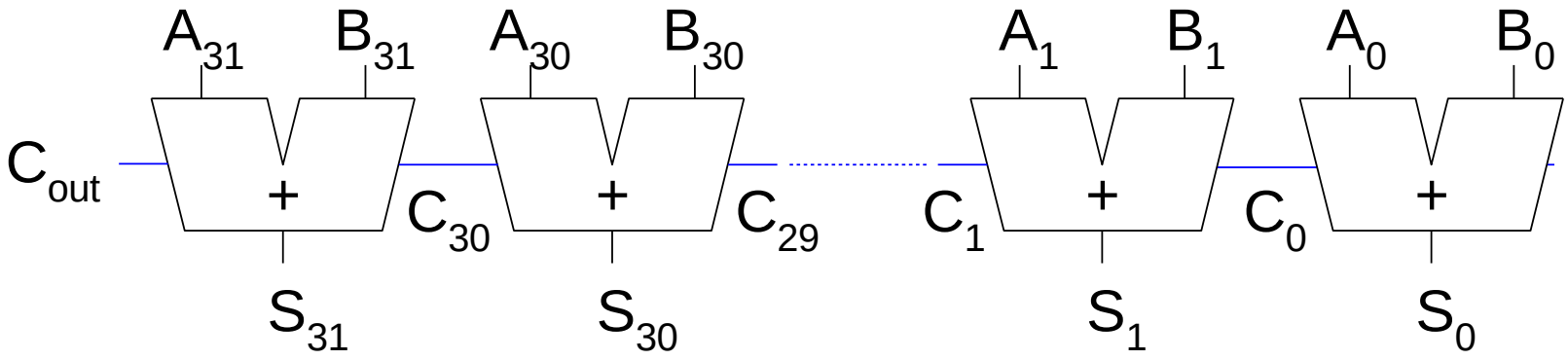


C.out has been rewritten to reduce the number of gates needed.

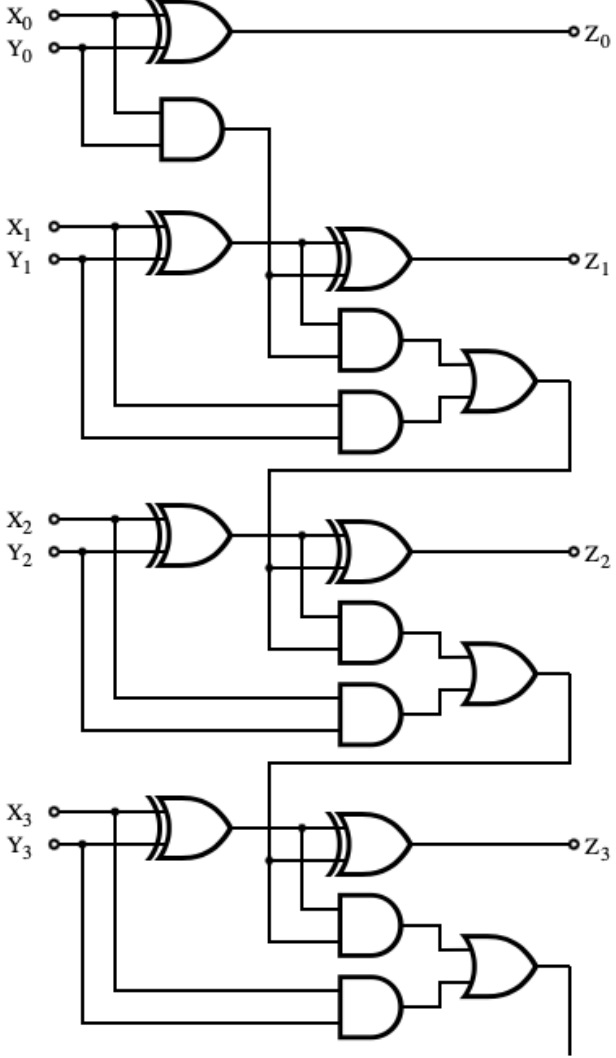
UNIVERSITY of VIRGINIA | ENGINEERING

RIPPLE CARRY ADDER

Next let's build a full adder



UNIVERSITY of VIRGINIA | ENGINEERING



RIPPLE CARRY ADDER

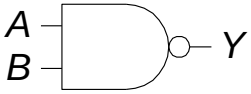
1 1 1 1 ← Carries

0 1 1 1

+1 0 1 1

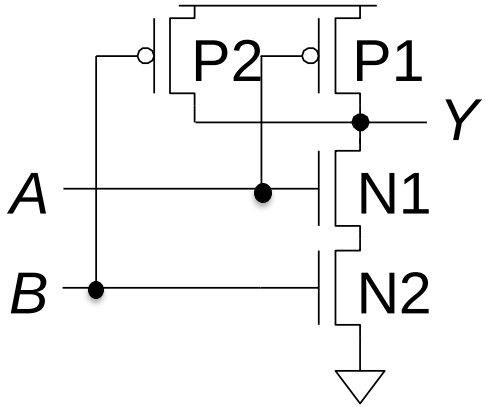
0 0 1 0

NAND



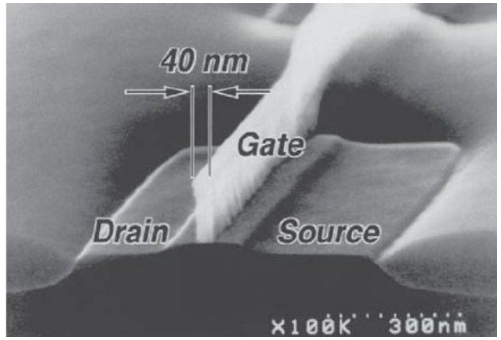
$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

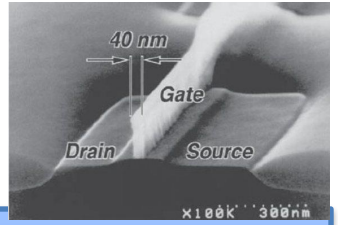


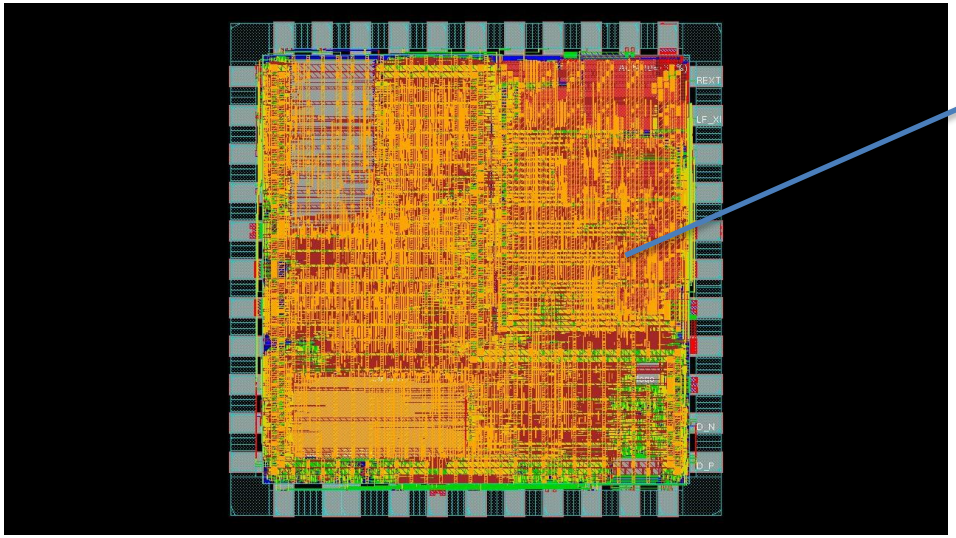
NAND gates are Turning complete you can build all other gates from them

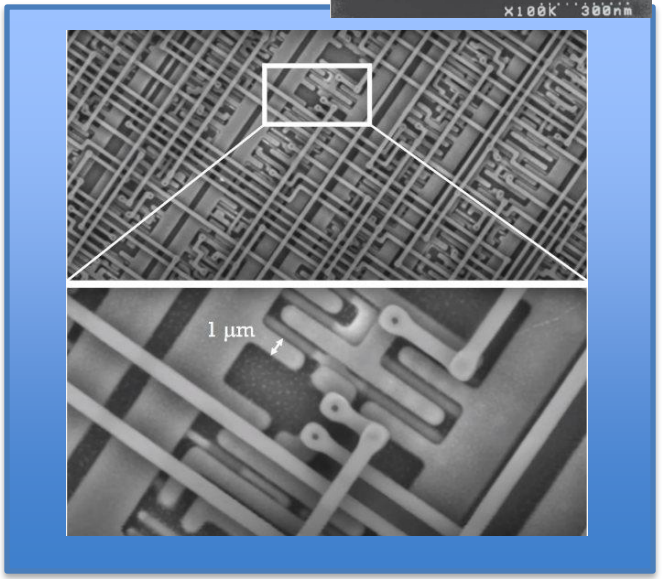
THIS IS WHERE WE'LL START OUR JOURNEY

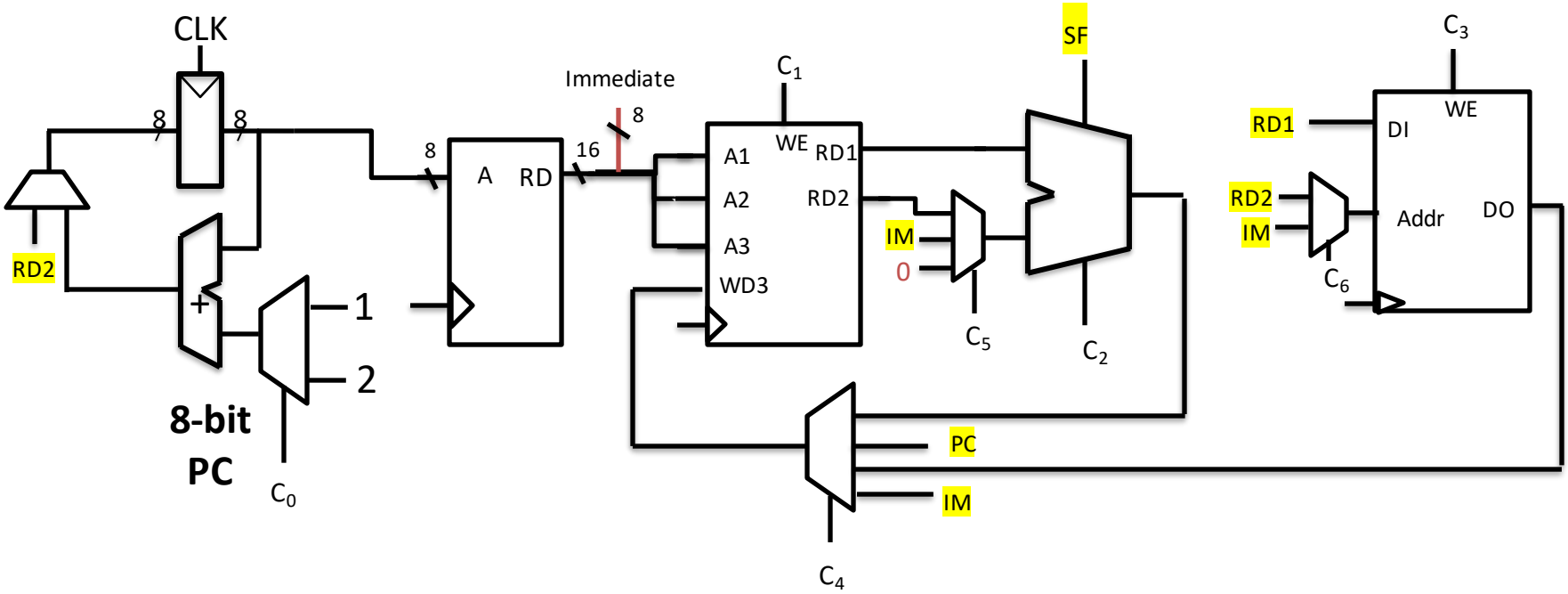


BOTTOM-UP APPROACH





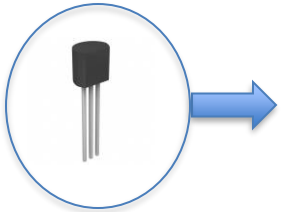


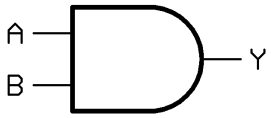


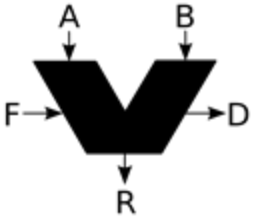
Write back stages

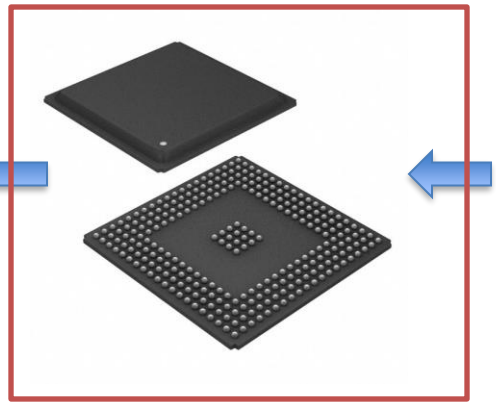
UNIVERSITY of VIRGINIA | ENGINEERING

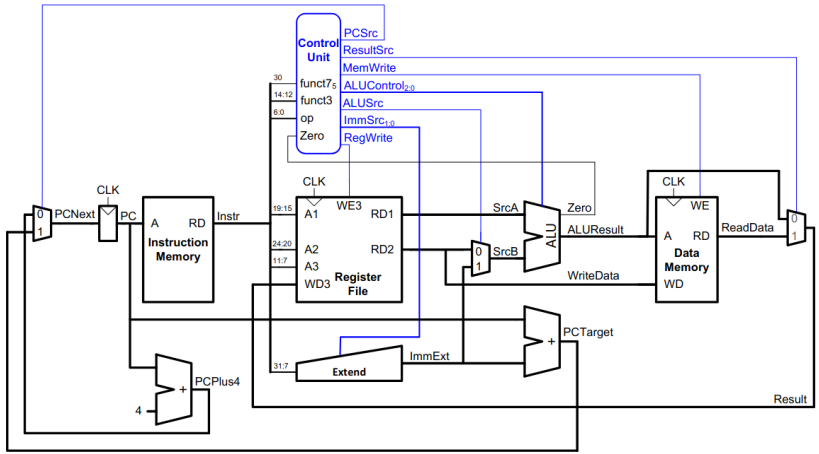
THE MAP (THE MACHINE)



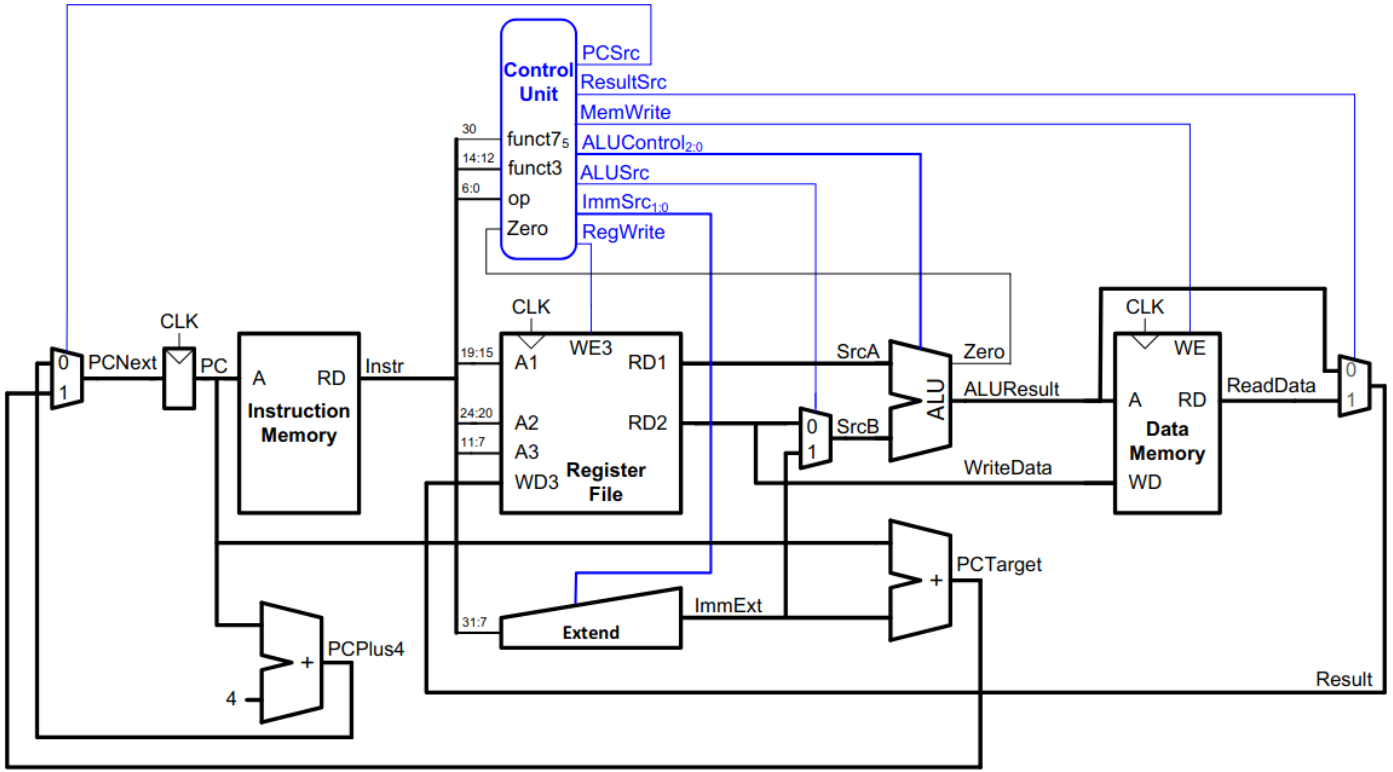








https://github.com/MKrekker/SINGLE-CYCLE-RISC-V



UNIVERSITY of VIRGINIA | ENGINEERING

WHAT ABOUT FABRICATING THESE

You can express our design in a programming language called VHDL.

Simulate your processor in model sim

And then send off the TSMC, UMC, or Samsung to get fabricated.

Don't worry you'll not have to write VHDL in this course. But ECE does offer courses. Maybe I will rework or simulation lab to give us a taste of this language.

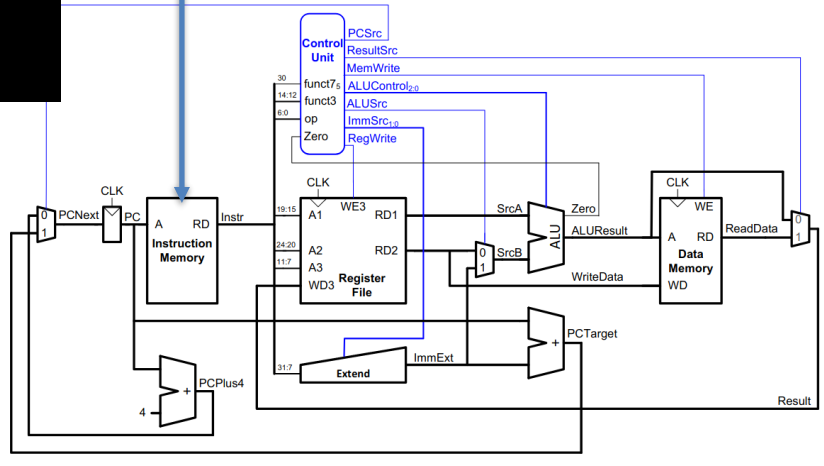
```
signal and_gate : std_logic;
and_gate <= input_1 and input_2;
```

```
entity example_and is
  port (
    input_1    : in  std_logic;
    input_2    : in  std_logic;
    and_result : out std_logic
  );
end example_and;
```

```
architecture rtl of example_and is
  signal and_gate : std_logic;
begin
  and_gate <= input_1 and input_2;
  and_result <= and_gate;
end rtl;
```

THE MAP (THE CODE)

```
0000000000001149 <main>:
   1149: f3 0f 1e fa        endbr64
   114d: 55                 push   %rbp
   114e: 48 89 e5           mov    %rsp,%rbp
   1151: 48 8d 05 ac 0e 00 00 lea    0xeac(%rip),%rax       # 2004
<_IO_stdin_used+0x4>
   1158: 48 89 c7           mov    %rax,%rdi
   115b: e8 f0 fe ff ff     call   1050 <puts@plt>
   1160: b8 00 00 00 00     mov    $0x0,%eax
   1165: 5d                 pop    %rbp
   1166: c3                 ret
```



UNIVERSITY of VIRGINIA | ENGINEERING

THE MAP (THE CODE)

```c
#include <stdio.h>
int main() {
        printf("Hello, World!");
        return 0;
}
```

```
0000000000001149 <main>:
   1149: f3 0f 1e fa        endbr64
   114d: 55                 push   %rbp
   114e: 48 89 e5           mov    %rsp,%rbp
   1151: 48 8d 05 ac 0e 00 00 lea    0xeac(%rip),%rax       # 2004
<_IO_stdin_used+0x4>
   1158: 48 89 c7           mov    %rax,%rdi
   115b: e8 f0 fe ff ff     call   1050 <puts@plt>
   1160: b8 00 00 00 00     mov    $0x0,%eax
   1165: 5d                 pop    %rbp
   1166: c3                 ret
```

We will not cover this conversion in detail. CS 4620 - Compilers is a class dedicated to building and understanding the program designed to do this conversion.

We'll focus on understanding the output of the program and how this output gets executed on a machine