CS-1000 An Introduction to Computer Architecture

Dr. Soner Onder

Michigan Tech

October 13, 2015

About Me

- BSc degree in Chemical Engineering from METU, Ankara, Turkey.
- MSc in Computer Engineering, METU, Ankara, Turkey.
- PhD in Computer Science, University of Pittsburgh, PA.
- Worked in industry both as a systems programmer, as well as a field engineer (8+ Years) before starting my Phd.
- Developed thousands of lines of code, most of which were utilized heavily.

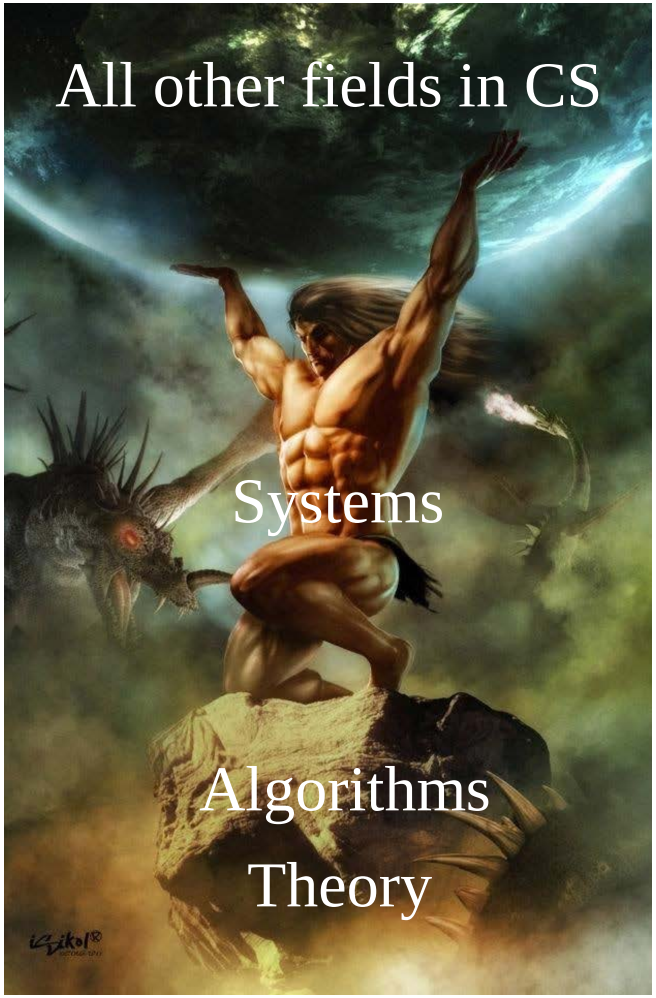
About Me

- Married to one of your professors.
- Have two kids, one is majoring in Chemical Engineering, the other is a sophomore in Calumet High School.

- Have a furry orange tabby cat
  - He probably is wearing a costume (not sure).
  - Acts like a black cat in Halloween.

That is not him !

My view of Computer Science



Without Algorithms and Theory there is NO Computer Science.

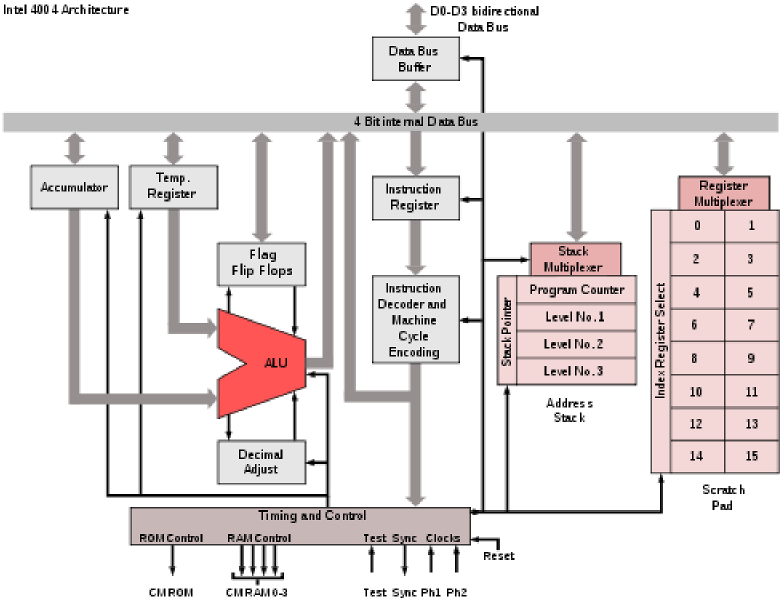
Without SYSTEMS, there is NO MACHINE (i.e., Computer).

Without COMPUTER, there is no **Smart Phone !**

What is **SYSTEMS?**

- □ **The core is Computer Architecture.**
- □ Programming Languages and Compilers.
- □ Operating Systems.
- □ Computer Networks.

Intel 4004 (1971)



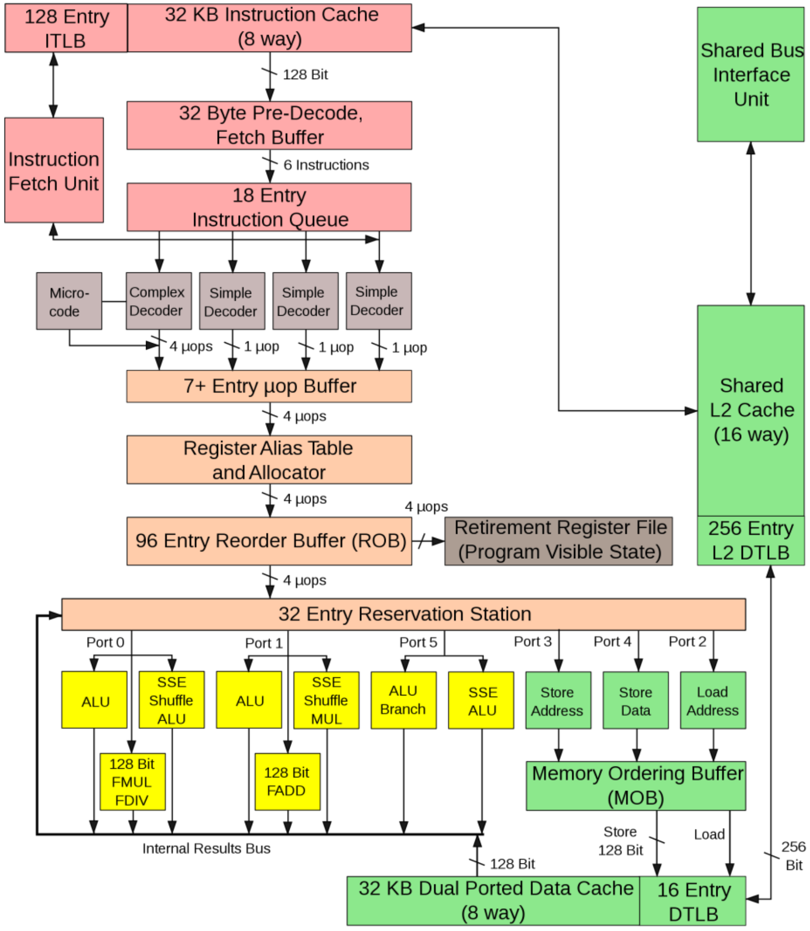
Maximum clock rate was 740 kHz.
Instruction cycle time: 10.8 µs.
(8 clock cycles / instruction cycle)
46300 to 92600 instructions per second.

Adding two 8-digit numbers (32 bits each, assuming 4-bit BCD digits) was stated as taking 850 µs - i.e. 79 instruction cycles, about 10 instruction cycles per decimal digit.

Instruction set contained 46 instructions (of which 41 were 8 bits wide and 5 were 16 bits wide)

Register set contained 16 registers of 4 bits each

Intel Core Architecture (2006)



Intel Core 2 Architecture

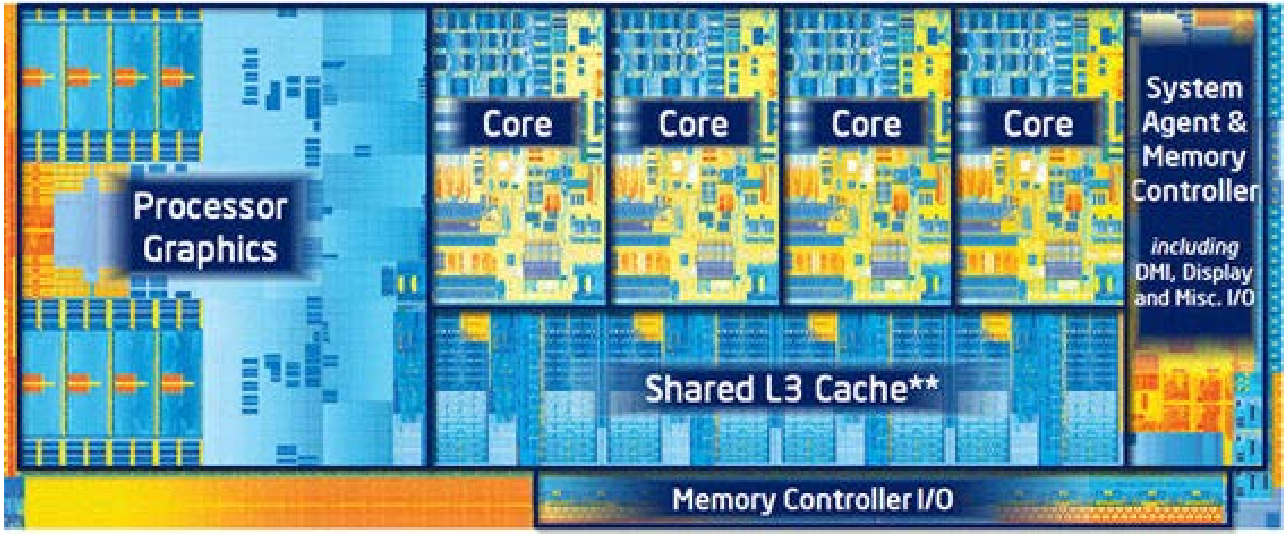
Clock rate 3GHZ.
6 - 9 Billion Instructions per second.

L1 cache 64 kB per core
L2 cache 1 MB to 8 MB unified
L3 cache 8 MB to 16 MB shared (Xeon)
Transistors 105M 65 nm

Intel Core i7 (2008)

Clock rate 3-3.5 GHZ.
6 -9 Billion Instructions per second/CPU.

Transistors
  730M 45 nm
  1.8 B (6 core – 2013)
  5.560 B (18-core Xeon Haswell- 2014)
Outlook:
  100 B transistors in 2020 !



Is Computer Architecture Circuits (Hardware)?

- No. But you need to understand how the hardware works.
- It is how we put together circuits (at a higher level of abstraction, and algorithmically):
  - Intel Core i7 (single core) / Intel 4004 =
  - 3,000,000,000 (Hz) / 740,000 (Hz) = 4054 times faster.
  - 6,000,000,000 (instructions/sec) / 92600 (instructions/sec) = 64,795
  - A factor of roughly 16 in performance !
- That is the power of computer architecture:
  - Modern processors process multiple instructions per cycle
  - They act speculatively to mitigate delays
  - They use sophisticated algorithms to efficiently execute programs.

Computer Architecture

Computer Architecture is a core field of computer science which sits at the cross-roads of abstractions.

- Very vibrant field – needs always changing together with opportunities.
  - New circuit techniques enable new architectures.
  - New architectures may facilitate new techniques.
- Optimize for power, performance (or both).

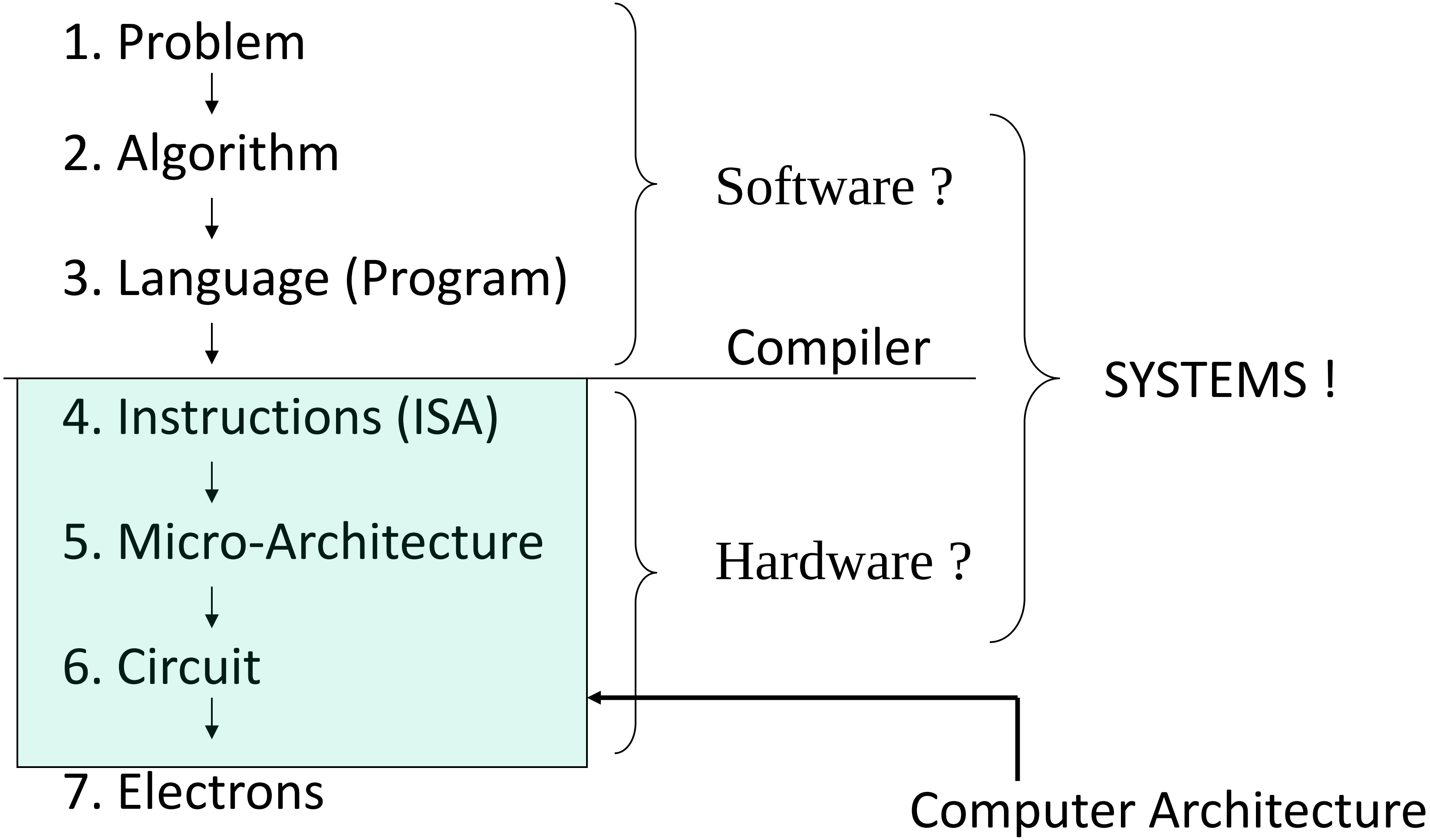
- Computer Architecture can potentially impact everything (yes, you can also save the world by being an architect!)
- Very high paying (and satisfying) good jobs too ..
  - Processors are everywhere from simple machines to war planes, from factories to kitchen appliances.

Revisiting Computer Science

- It is the science of creating and utilizing abstractions to achieve computation.
- Using abstractions is the only way we know to create complex systems.

- **Computer Architecture is a core field of computer science which sits at the cross-roads of abstractions.**
  - *Only if you learn and understand all the layers we use in computation you can become a good architect.*

Layers of Abstractions



1. Problem

2. Algorithm

3. Language (Program)

4. Instructions (ISA)

5. Micro-Architecture

6. Circuit

7. Electrons

Software ?

Compiler

Hardware ?

SYSTEMS !

Computer Architecture

My Research

- Primarily concentrated on three fronts :
    1. Seeking alternative forms of execution models so that sequential programs can be executed efficiently by highly parallel architectures.
    2. Dealing with latency/delays : Seeking ways to execute dependent instructions together.
    3. Applying AI techniques on Computer Architecture, primarily on simulators to verify their correctness and further understand behavior of complex architectures.

Project Sphinx

©Glenn and Gary McCoy/Distributed by Universal Uclick via CartoonStock.com
7/22
Gary
CARTOONSTOCK .com
Search ID: ggm090722
"ON SECOND THOUGHT, LET'S TRY IT THE OTHER WAY AROUND... MY HEAD ON A LION'S BODY."

This is a joint four year project between MTU and FSU

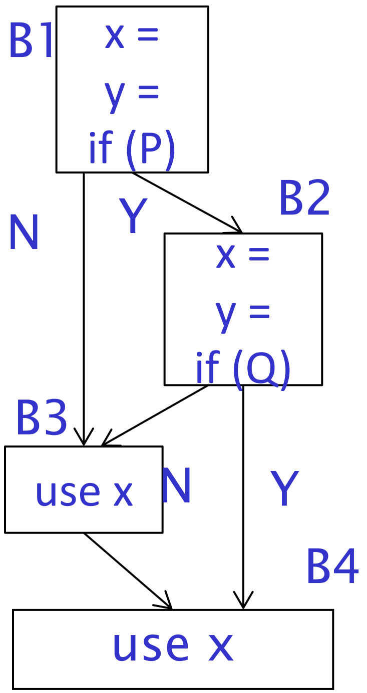
(Co-PIs : Soner Onder and David Whalley)

Funded by NSF ($745,000, MTU Share $560,000, MTU is the lead institution).

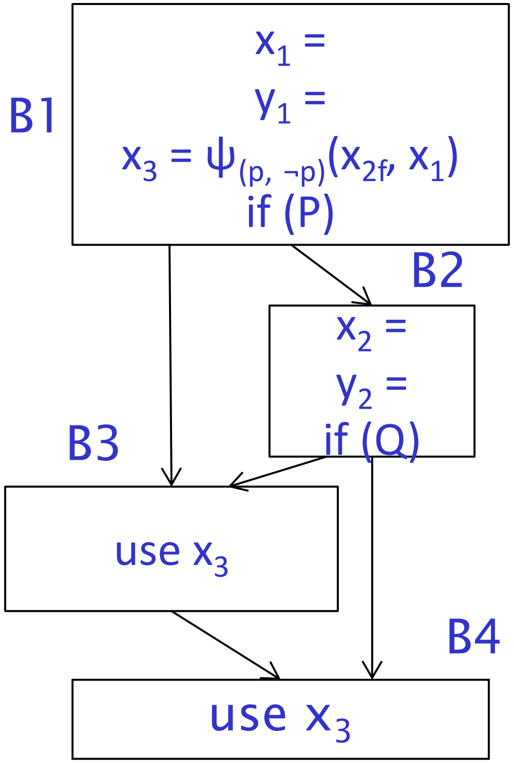
Project Goals:

- Exploit both regular and irregular parallelism.
- Massive ILP through LaZy execution.
- Imperative programming languages by translating to FGSA.
- Single-assignment form for both the compiler and the architecture.
- Multi-core uniprocessor!

An FGSA Example



original program



FGSA

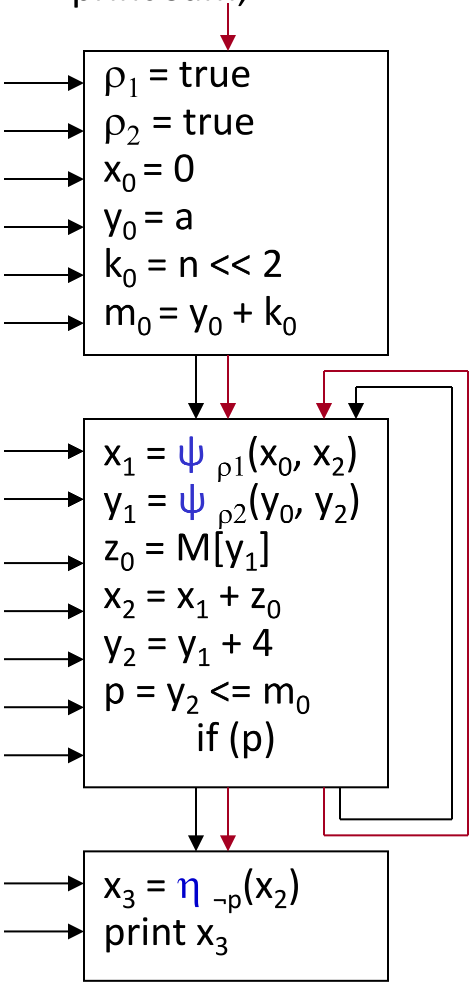
Algorithms for converting programs into FGSA:

Dr. Shuhan Ding
PhD in 2012
Michigan Tech

$CC = <<\{B1.x,B2.x\}, \{B3.x,B4.x\}>, \{p, \neg p\},\psi>$

Supporting Execution Models - the old shoe

```
do i=0 step 1 until n
sum = sum + a[i];
print sum;
```



ρ_1 = true
ρ_2 = true
$x_0 = 0$
$y_0 = a$
$k_0 = n << 2$
$m_0 = y_0 + k_0$

$x_1 = \psi_{\rho 1}(x_0, x_2)$
$y_1 = \psi_{\rho 2}(y_0, y_2)$
$z_0 = M[y_1]$
$x_2 = x_1 + z_0$
$y_2 = y_1 + 4$
$p = y_2 <= m_0$
if (p)

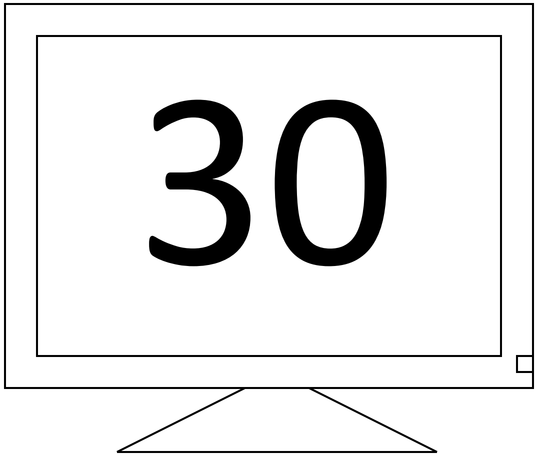
$x_3 = \eta_{\neg p}(x_2)$
print x_3

n	a[0]	a[1]
1	10	20

ρ_1	ρ_2	x_0	y_0	k_0	m_0
true false	true false	0	a	4	a+4

x_1	y_1	z_0	x_2	y_2	p
0 10	a a+4	10 20	10 30	a+4 a+8	true false

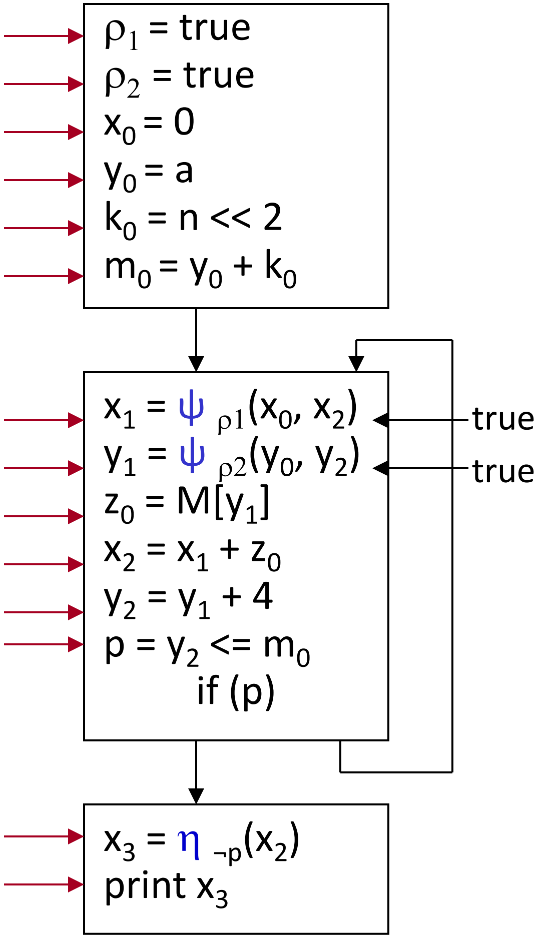
x_3
30



Supporting Execution Models – demand driven execution

```
do i=0 step 1 until n
sum = sum + a[i];
print sum;
```

n	a[0]	a[1]
1	10	20



ρ_1 = true
ρ_2 = true
$x_0 = 0$
$y_0 = a$
$k_0 = n << 2$
$m_0 = y_0 + k_0$

$x_1 = \psi_{\rho 1}(x_0, x_2)$
$y_1 = \psi_{\rho 2}(y_0, y_2)$
$z_0 = M[y_1]$
$x_2 = x_1 + z_0$
$y_2 = y_1 + 4$
$p = y_2 <= m_0$
if (p)

$x_3 = \eta_{\neg p}(x_2)$
print x_3

ρ_1	ρ_2	x_0	y_0	k_0	m_0
false	false	0	a	4	a+4

x_1	y_1	z_0	x_2	y_2	p	x_3
0	a	10	10	a+4	true	

Demand	Execute
x_3	
p, x_2	
y_2,m_0,x_1,z_0	
y_1,y_0,k_0,ρ_1	
ρ_2	ρ_1,k_0,y_0
x_0	ρ_2
	x_0,m_0,y_1
	z_0,y_2,x_1
	p,x_2

End of first iteration.
η sees that ¬ p is false and demands both p and x_2 again.