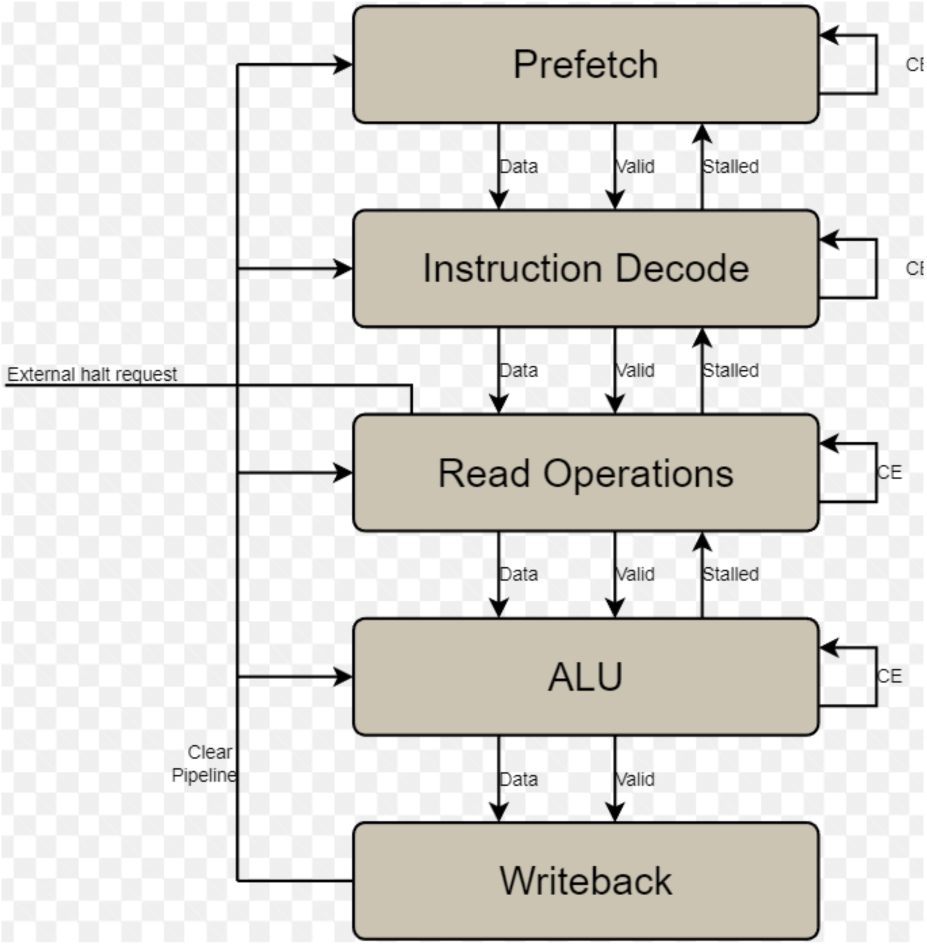
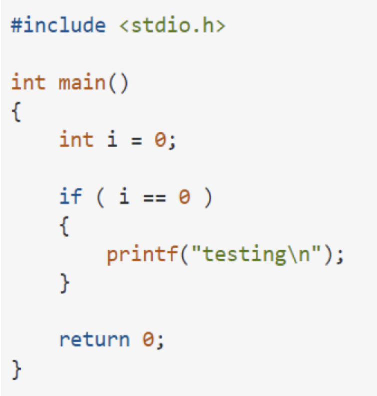
CAS CS 440
Lec 35

Assembly Calculus

1. What is Assembly Calculus?
    a. Observation from Computers
        i. CPU is the “brain” of the computer
            1. Designed as a pipeline (for efficiency)
            2. Can only do a few things
                a. X86_64: 981 unique mnemonics
                b. Mips: 47 unique mnemonics
                c. A64 (aarch64): 442 unique mnemonics
    b. Using those mnemonics (instructions):
        i. Can write any program we wish
        ii. Languages are Turing complete: any computation program
    c.



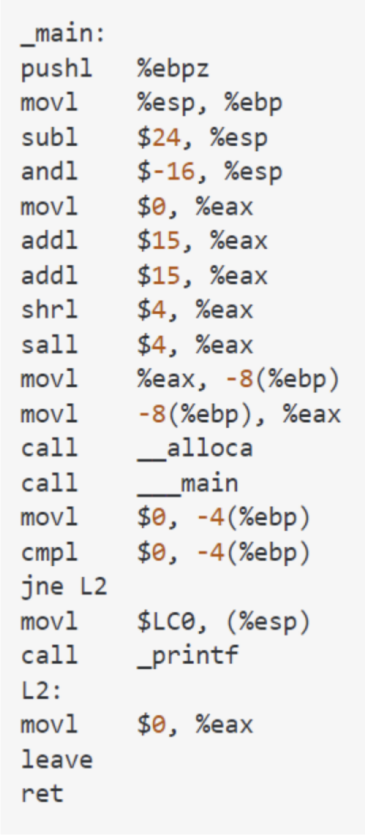
d. 

```c
#include <stdio.h>

int main()
{
    int i = 0;

    if ( i == 0 )
    {
        printf("testing\n");
    }

    return 0;
}
```

e. 

```
_main:
pushl   %ebpz
movl    %esp, %ebp
subl    $24, %esp
andl    $-16, %esp
movl    $0, %eax
addl    $15, %eax
addl    $15, %eax
shrl    $4, %eax
sall    $4, %eax
movl    %eax, -8(%ebp)
movl    -8(%ebp), %eax
call    __alloca
call    ___main
movl    $0, -4(%ebp)
cmpl    $0, -4(%ebp)
jne L2
movl    $LC0, (%esp)
call    _printf
L2:
movl    $0, %eax
leave
ret
```

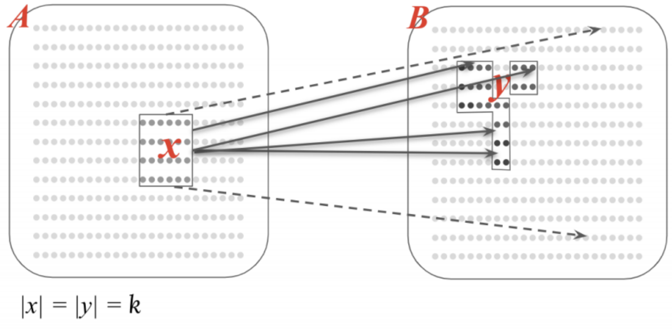
f. The downside:
   i. Complex programs:
      1. long sequences of primitives
   ii. Compiler!

g. The observation:
   i. Complex behavior = assembly of primitive behavior
   ii. Is this what our brain is doing?

h. What is the primitive operation in the brain?
   i. Is it action potentials (neurons firing)
   ii. Is it groups of neurons firing together?
   iii. etc

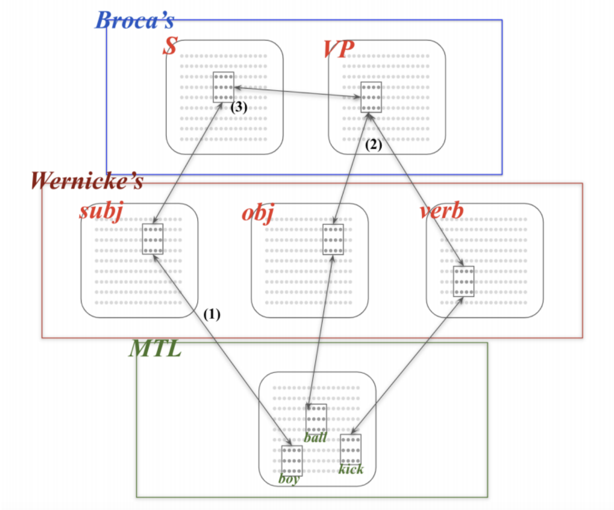
        i. Assembly calculus:
            i. Interested in how neurons (units) fire together
                1. Firing is the behavior that encodes information (i.e. the data)
                2. What kind of relationships neurons form = primitives (i.e. the instruction set)

2. Takeaways
    a. Composition of assemblies underly intelligence
    b. Dynamic Topologies
    c. Hebbian learning (biologically plausible) provides convergence
        i. Hebbian learning:
            1. Strengthen relationship between two units if their firings are correlated
            2. "Those that fire together wire together"
3. The Core Hypothesis
    a. Assembly Hypothesis
        i. Intelligence arises from the composition of primitive computing units
        ii. Assembly = graph of units (Erdos-Renyi)
        iii. Operations = modify existing assemblies / create new assemblies
        iv. Hebbian learning ("fire together → wire together")
            1. When two units fire at the same time, increase strength of connection
            2. When two units don't fire at same time, decrease strength of connection
        v. *k* units fire at a time (within a brain area)
4. The Core Hypothesis: Operations
    a. Projection :
        i. Copy an assembly *x* from area *A* to area *B* (new assembly called *y*)
            1. *y* will fire whenever *x* fires (*B* is downstream of *A*)
            2. When *x* fires, it excites units in *B*, if *x* keeps firing, different sets of units in *B* fire
                a. Process converges exponentially fast (from hebbian learning) to set *y*
    b.



- c. Association:
  - i. Link two assemblies together
    1. Observation: neuron fires when shown image of pyramid
    2. Shown image of person next to pyramid
    3. Neuron now fires when shown image of person!
    4. Neuron now belongs to multiple assemblies
  - ii. Assemblies are associative when units migrate between the assemblies
    1. Same brain area
    2. Overlap is preserved in projected assemblies
- d. Pattern Completion:
  - i. Whole assembly fires when a small number of its units fire
- e. Merge:
  - i. Create new assembly *z* in area *A* with strong two-way synaptic connectivity with assemblies *x* and *y* (in different brain areas).
  - ii. Unique to humans?
- f. Reciprocal Project:
  - i. The project operation but with strong backward synaptic connectivity
- g. RP&C:
  - i. Random synaptic connectivity between (and within) populations, and selecting (through inhibition) of *k* units to fire.

5. Hierarchical Processing
   - a. *Merge*, *project*, *reciprocal project* allow trees (of assemblies) to form
     - i. Recursive structure
     - ii. Process hierarchical information
       1. Language
       2. Images
       3. etc
   - b. Generating Sentences:
     - i. Modeled as PCFGs (hierarchical)
     - ii. Ex: “boy kicks ball”



c. Figure 2: A potential architecture for syntax in the brain.

6. Summary

   a. Composition of assemblies underly intelligence

   b. Operations must produce new assemblies/modify existing ones

      i. Topology is not static!

   c. Hebbian learning provides convergence