Simple as possible (SAP) Computer: Instruction Set & Programming

Spring 2023

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

Agenda

- Announcements (~ 5 mins)
- Poll everywhere (~10 mins)
- Language Concepts
- SAP Assembly Programming
- SAP ALU Flags
- SAP Jump instructions

(~60 mins)

Announcements

Lab 3

- Released yesterday (2/15)
- Due Feb. 24th @ 11:55 pm with no late penalty.

Quiz 3

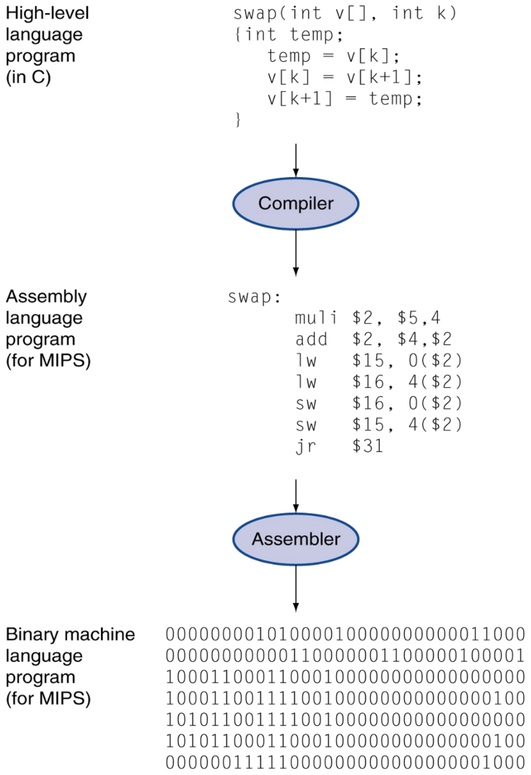
- Released this Friday
- Detailed information provided in Sakai announcement

Exam 1

- Grades and solution released yesterday (2/15)
- Regrade request begins today (2/16) @ 9:00 am and ends Sunday (2/19) @ 9:00 am.

Language Concepts

Assembly and Machine



High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Assembly: Review

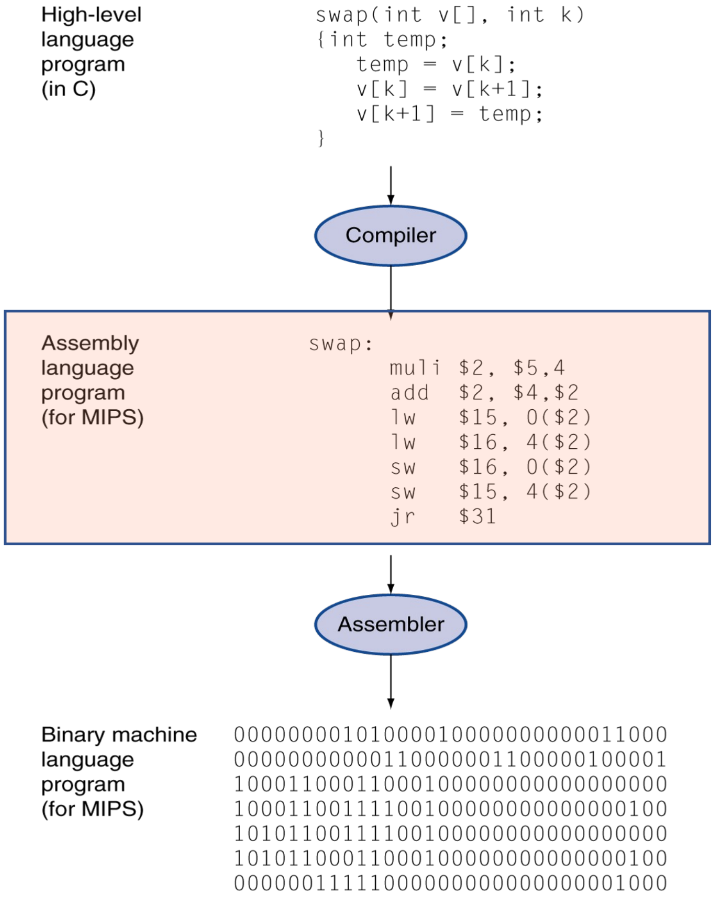
Assembly Program

- Created by the compiler
- One or more **assembly instructions** that perform a specific task
- Human readable textual representation

Assembly Instruction

- Arithmetic, memory, logic, control, etc.
- **Instruction set architecture** is the full vocabulary that combines instructions with registers, addressing modes, data types
- Processor architectures (RISC and CISC)
- RISC processor: MIPS, ARM, RISC-V, etc.
- CISC processor: x86

For humans (not machines)


High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Machine: Review

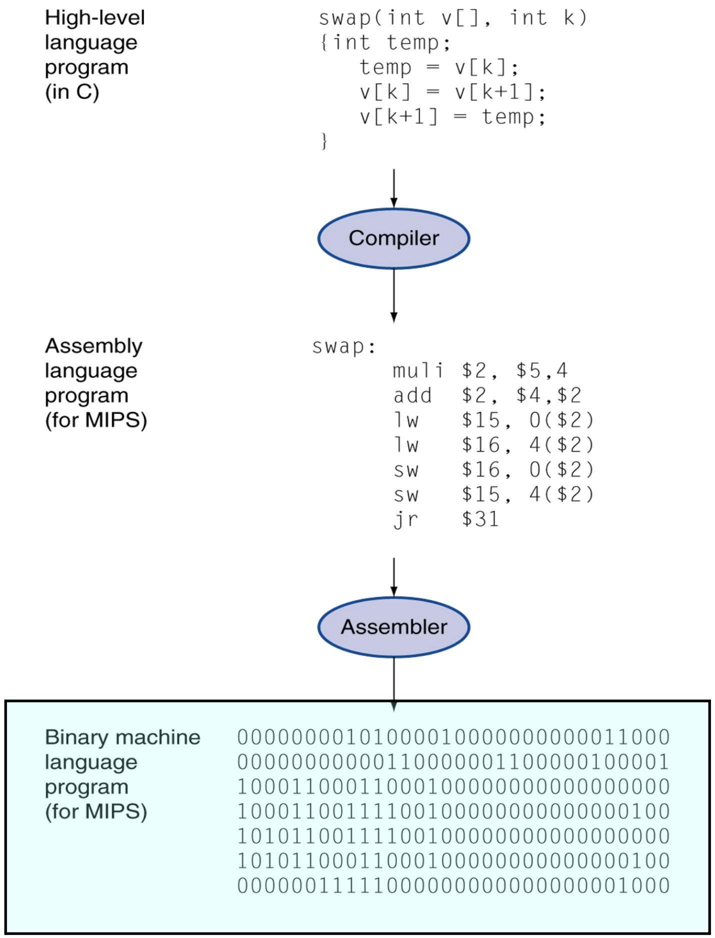
Program

- One or more **machine instructions** that perform a specific task
- Created by the assembler
- Not human readable binary representation

Instructions

- One-to-one correspondence with assembly instruction
- Specific to processor architecture (hardware)!

For machines (not humans)



High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

SAP ALU

Design and Operation

Condition Flags

Z (zero)

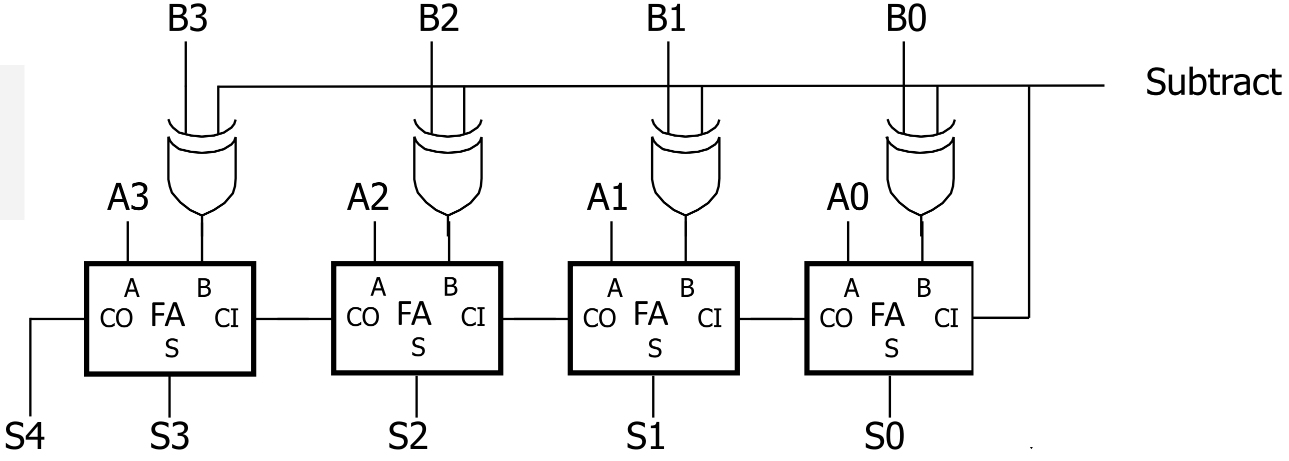
- S3, S2, S1, S0 are all 0

C (carry-out)

- S4 is 1
- Most significant bit (MSB) produced a carry-out

Revisit Add/Subtract circuit

- 4-bit example
- S4 = carry-out bit



Comparing Unsigned Numbers

To compare A and B:

- First compute A – B
- Then check ALU flag bits Z and C

Examples:

- LTU (less than for **unsigned** numbers)
- A < B is given by ~C
- Others in table

Unsigned comparison:

EQ	==	Z
NE	!=	~Z
LTU	<	~C
LEU	<=	~C+Z
GEU	>=	C
GTU	>	~(~C+Z)

ATTENTION PLEASE

SAP does not support less than (LT) signed number comparisons!

Example: Comparing Numbers

A = 0001_2 B = 0010_2

A = 0010_2 B = 0001_2

Flag Register and Operations

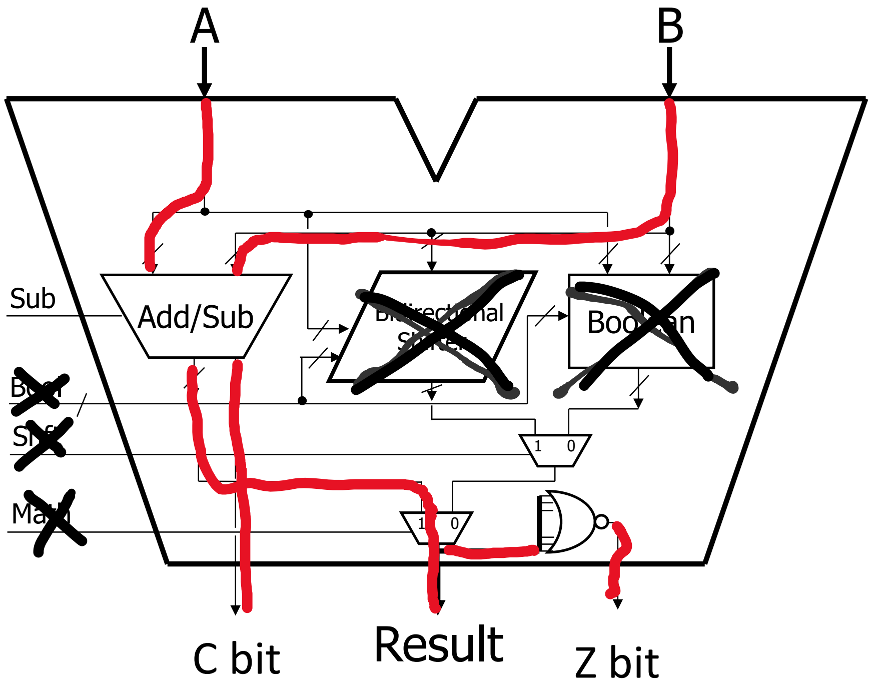
Our ALU design with Z and C flags Note: SAP does not include Shift and Boolean operations

2-bit flag register

C-bit	Z-bit

C = 1: carry-out bit = 1
C = 0: carry-out bit = 0

Z = 1: result = 0
Z = 0: result != 0



SAP Assembly Programming

Instruction Set

Instruction Mnemonic

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

Three or two letter acronym that represents a binary instruction.

- SAP has 11 instructions in total (OUT will not be used)

Binary Opcode

4-bit representation of mnemonic

- Must be unique, i.e., two or more opcode values cannot be the same!
- Every instruction must have an opcode!

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

These bits are used by control

- Configure SAP data-path (i.e., BUS, registers, ALU, etc.)
- More to come!!

If an opcode is 4-bits, then what is the maximum number of instructions that can be supported?

Arg (or Operand)

Instruction **arg**ument

- Typically, base-10 or base-16 representation
- For example, `LDA 3`, where Arg=3

Usage:

- Address in RAM that stores data (i.e., data address)
- Address in RAM that stores an instruction (i.e., instruction address)
- Immediate (constant) data
- Nothing (NOP, OUT, HLT)

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

LDA Function

`A = mem(arg)`

means,

At address `arg` in RAM load data in to register `A`.

Textual meaning of instruction

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

ADD Function

`A = A + mem(arg)`

means,

At address `arg` in RAM load data in to register `B`. Then perform ALU operation `A = A + B`

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

ATTENTION PLEASE

SUB instruction is very similar, the only difference is ALU subtraction, i.e., A = A - B

STA Function

`mem(arg)= A`

means,

At address `arg` in RAM store the data held in register `A`.

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

LDI Function

A = arg

means,

Set the value in register A to the arg immediate value.

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

JMP Function

`PC = arg`

means,

Set the value in register `PC` to the `arg` immediate value.

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

JC (Jump Carry-out) Function

```
if FC=1,
        PC = arg
else
        PC = PC + 1
```

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

means,

If the ALU flag carry-out bit (FC) is 1,

Set the value in register PC to the arg immediate value.

else,

increment the value in the PC register by 1

JZ (Jump Zero) Function

```
if FZ=1,
     PC = arg
else
     PC = PC + 1
```

Op Code	Instruction	Arg	Function
0000	NOP		
0001	LDA	Data Address	A = Mem(arg)
0010	ADD	Data Address	A = A + Mem(arg)
0011	SUB	Data Address	A = A - Mem(arg)
0100	STA	Data Address	Mem(Arg) = A
0101	LDI	Constant value	A = arg
0110	JMP	Instruction Address	PC = arg
0111	JC	Instruction Address	if FC=1 then PC=arg else go on
1000	JZ	Instruction Address	if FZ=1 then PC=arg else go on
1110	OUT		Display = OUT = A
1111	HLT		

means,

If the ALU flag zero bit (FZ) is 1,

Set the value in register PC to the arg immediate value.

else,

increment the value in the PC register by 1

SAP Assembly Programming

Jump instructions

Two Types of Jump Instructions

Unconditional jump:

- JMP instruction
- Procedure/function call
- Loop flow control (i.e., repeat loop)

Conditional jump:

- JC and JZ instructions
- Branch statements
  - if-else
  - loops (for, while, do-while)
- Condition is true (FC or FZ = 1) take the branch
- Condition is false (FC or FZ = 0) don’t take the branch

```
Pseudo-code

func() {
   A = A + A
}
```

```
if ( A < B )
   …
else
   …
```

Example: JMP Instruction

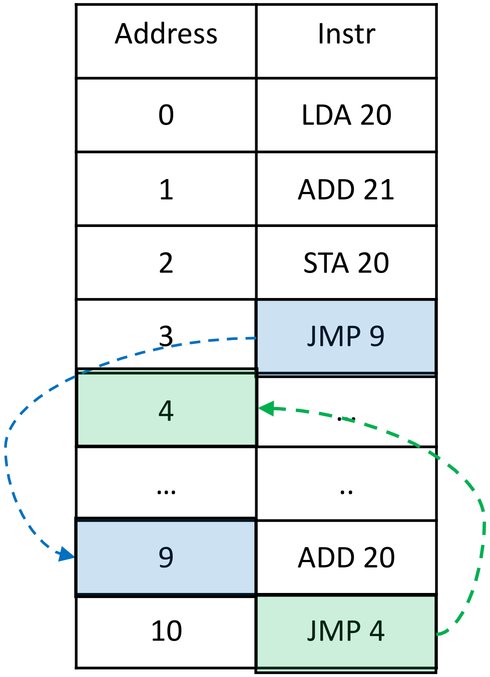
Pseudo-code

```
A = A + B
func()
…

func() {
    A = A + A
}
```

1. Execute instruction address 0
2. Execute instruction address 1
3. Execute instruction address 2
4. Execute instruction address 3
   - Jump to address 9
5. Execute instruction address 9
6. Execute instruction address 10
   - Jump to address 4

Memory Abstraction



Address	Instr
0	LDA 20
1	ADD 21
2	STA 20
3	JMP 9
4	…
…	..
9	ADD 20
10	JMP 4

Assume:

- Variable A @ address 20
- Variable B @ address 21

Example: JZ Instruction

Pseudo-code

```
if ( A != B ) {
    A = A + A
}
…
```

If SAP FZ flag is 1, then branch because A == B.

1. Execute instruction address 0
2. Execute instruction address 1
3. Execute instruction address 2
   - if FZ = 1, jump (branch) to address 5 and execute instruction
   - else (no branch) execute instructions at address 3 and 4

Memory Abstraction

Address	Instr
0	LDA 20
1	SUB 21
2	JZ 5
3	LDA 20
4	ADD 20
5	…

Assume:
- Variable A @ address 20
- Variable B @ address 21

Example: JC Instruction

Pseudo-code

```
if ( A < B ) {
    A = A + A
}
…
```

SAP FC flag is opposite of ~C. Specifically, if FC=1 then ~C is 0.

1. Execute instruction address 0
2. Execute instruction address 1
3. Execute instruction address 2
   - if FC = 1, jump (branch) to address 5 and execute instruction
   - else (no branch) execute instructions at address 3 and 4

Memory Abstraction

Address	Instr
0	LDA 20
1	SUB 21
2	JC 5
3	LDA 20
4	ADD 20
5	…

Assume:
- Variable A @ address 20
- Variable B @ address 21