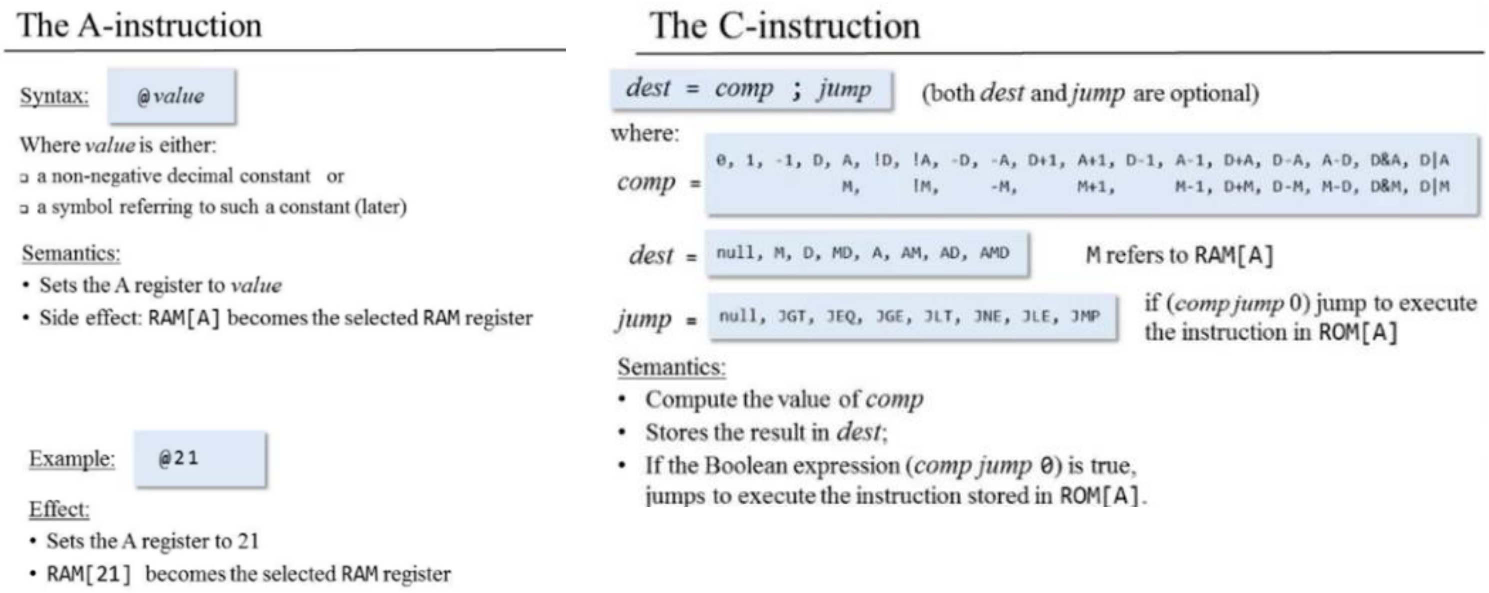
Lab #5 - Machine Language Basics

Name:______________________

Section/Time:______________________

Date: ______________________

Recall the two Assembly Instructions, A and C:



The A-instruction

Syntax: `@value`

Where *value* is either:

- a non-negative decimal constant or
- a symbol referring to such a constant (later)

Semantics:

- Sets the A register to *value*
- Side effect: RAM[A] becomes the selected RAM register

Example: `@21`

Effect:

- Sets the A register to 21
- RAM[21] becomes the selected RAM register

The C-instruction

dest = *comp* ; *jump* (both *dest* and *jump* are optional)

where:

comp = 0, 1, -1, D, A, !D, !A, -D, -A, D+1, A+1, D-1, A-1, D+A, D-A, A-D, D&A, D|A
M, !M, -M, M+1, M-1, D+M, D-M, M-D, D&M, D|M

dest = null, M, D, MD, A, AM, AD, AMD M refers to RAM[A]

jump = null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP if (*comp jump* 0) jump to execute the instruction in ROM[A]

Semantics:

- Compute the value of *comp*
- Stores the result in *dest*;
- If the Boolean expression (*comp jump* 0) is true, jumps to execute the instruction stored in ROM[A].

Translate the following into Assembly Instructions:

1) Set RAM[0] to 3
Set RAM[1] to 5
Set RAM[2] to 1
Set RAM[3] to -1
	@3 @5 @2 @3
	D=A D=A m=1 m=-1
	@0 @1
	m=D m=D

2) Set RAM[0] to 2
Set RAM[1] to 3
Set RAM[2] = RAM[0] + RAM[1]
	@2 @3
	D=A D=A
	@0 @1
	m=D m=D

3) Set **D** to **A - 1**	D=A-1
4) Set both **A** and **D** to **A + 1**	AD=A+1
5) Set **D** to **19**	@19 D=A

6) Set both **A** and **D** to **A + D**	AD=A+D
7) Set **RAM[5034]** to **D - 1**	@5034 M=D-1

8) Set **RAM[543]** to **171**	@171
	D=A
	@543
	M=D

9) Increment **RAM[7]** by **1** and store result in **D**	@7 D=M+1

10) Increment **RAM[12]** by **3** and store result in **D**	@3
	D=A
	@12
	D=D+M

11) // Convert the following Java code to assembly
`int i = 5;`
`i++;`
`i+=2;`
`i-=3;`	

12) // Convert the following Java code to assembly
`int i = 5;`
`int j = 10;`
`int k = i - j;`	

Translate the following tasks into Assembly Instructions

1) `sum = 0`	@sum M=0
2) `j = j + 1`	@j M=M+1
3) `q = sum + 12 - j`	@sum D=M @12 D=D+A @j D=D-M @q

4) // Declare that arr=100 and n =10
`int n = 10;`
`int[] arr = new int[n];`
`arr[3] = -1`	

5) // Assume that j has already been declared
`arr[j] = 0`
	@j
	D=M
	@arr
	A=D+M
	M=0

6) `arr[j] = 17`	@j
	D=M
	@arr
	D=D+M
	@Ptr
	M=D
	@17
	D=A
	@PtR
	A=M
	M=D

Lab #5 - Machine Language Jumps

Translate the following instructions into Assembly Instructions

1) `goto 50`	
2) `if D==0 goto 112`	
3) `if D<9 goto 507`	
4) `if RAM[12]>0 goto 50`	
5) `if sum>0 goto END`	
6) `if x[i]<=0 goto NEXT`	

Lab #5 - Machine Language Loops

Translate the following instructions into Assembly Instructions

1)

```
int n = 5;
for (int i=1;i<=n;i++) {}
```

2)

```
int sum = 0;
int n = 5;
for (int i=1;i<=n;i++) {
  sum += i;
}
```

3)

```
// Declare an arr at RAM[20]
// Size (n) of 10
for (int i=0; i<n; i++)
   arr[i] = -1;
```

4)

```
// Declare an arr at RAM[20]
// Size (n) of 5
for (int i=0; i<n; i++)
   arr[i] = 100;
```