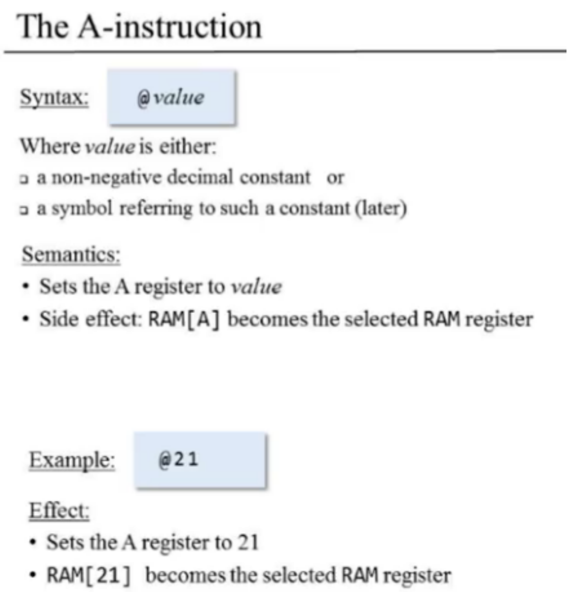
Lab #5 - Machine Language Basics

Name: Raul Aguilar
Section/Time: CS 220 2148
Date: October 1, 2020

Recall the two Assembly Instructions, A and C:



The A-instruction

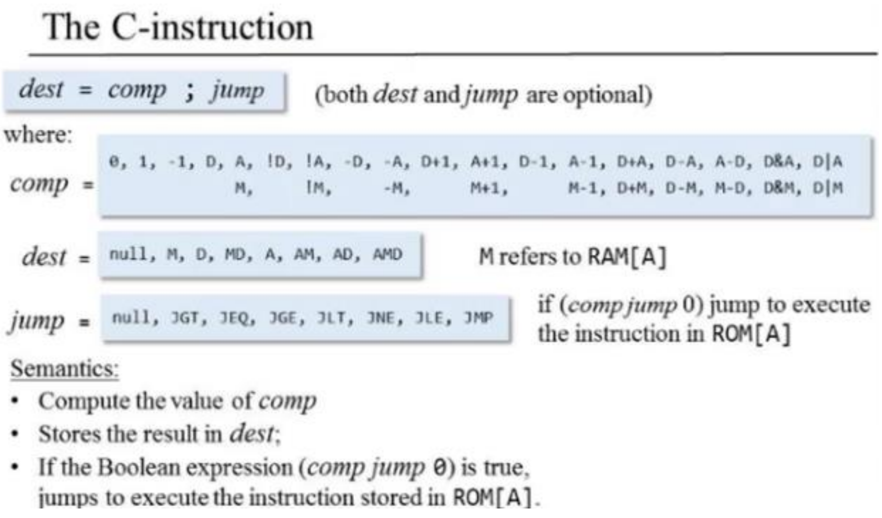
Syntax: `@value`

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
	D = A
	@0
	M = D
	@5
	D = A
	@1
	M = D
	@2
	M = 1
	@3
	M = -1

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

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

11) // Convert the following Java code to assembly
`int i = 5;`
`i++;`
`i+=2;`
`i-=3;`
	@5
	D=A
	@i
	M=D
	M=M+1
	@2
	D=A
	@i
	M=M+D
	@3
	D=A
	@i
	M=M-D

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

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`	@12
	D=A
	@sum
	D=D+M
	@j
	D=D-M
	@q
	M=D

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

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

6) `arr[j] = 17`	

Lab #5 - Machine Language Jumps

Translate the following instructions into Assembly Instructions

1) `goto 50`	@56 0;JMP
2) `if D==0 goto 112`	@112 D;JEQ
3) `if D<9 goto 507` D-9<0	@9 D=D-A @507 D;JLE
4) `if RAM[12]>0 goto 50`	@12 D=M @50 D;JGT
5) `if sum>0 goto END`	@sum D=M @END D;JGT
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++) {}`
i-n<=0
	@5
	D=A
	@n
	M=D
	(FOR)
	@i
	M=1
	D=M
	@n
	D=D-M
	@ENDFOR
	D;JLE
	@i
	M=M+1
	@FOR
	0;JMP
	(ENDFOR)

2)
`int sum = 0;`
`int n = 5;`
`for (int i=1;i<=n;i++) {`
`  sum += i;`
`}`
	@sum
	M=0
	@5
	D=A
	@n
	M=D
	(FOR)
	@i
	M=1
	D=M
	@n
	D=D-M
	@ENDFOR
	D;JLE
	@i
	D=M
	@sum
	M=M+D
	@i
	M=M+1
	@FOR
	0;JMP
	(ENDFOR)

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;
```