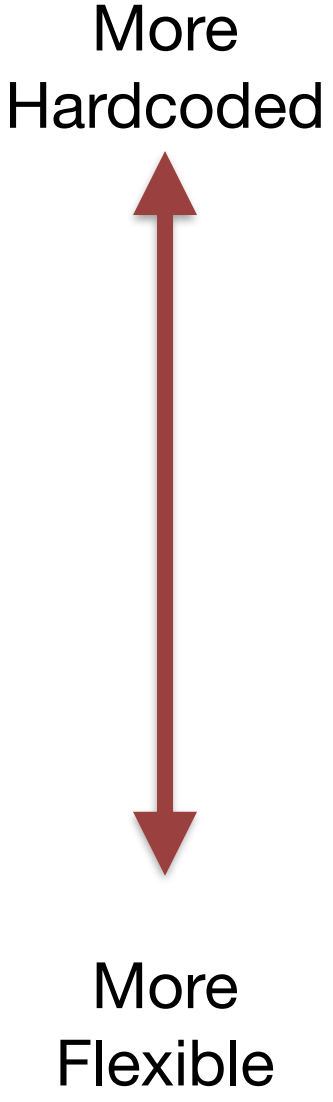
ADDRESSING MODES

Immediate	Number hardcoded into instruction	`MOVE.L #5,D0`
Direct	Address hardcoded into instruction	`MOVE.L VAR,D0`
Register Indirect	Address comes from an address register	`MOVE.L (A0),D0`
Register Indirect + Postincrement	Address comes from an address register, which is incremented after the instruction completes	`MOVE.L (A0)+,D0`
Register Indirect + Index	Address comes from the sum of address register and data register	`MOVE.L (A0,D1),D0`



IMMEDIATE

Java

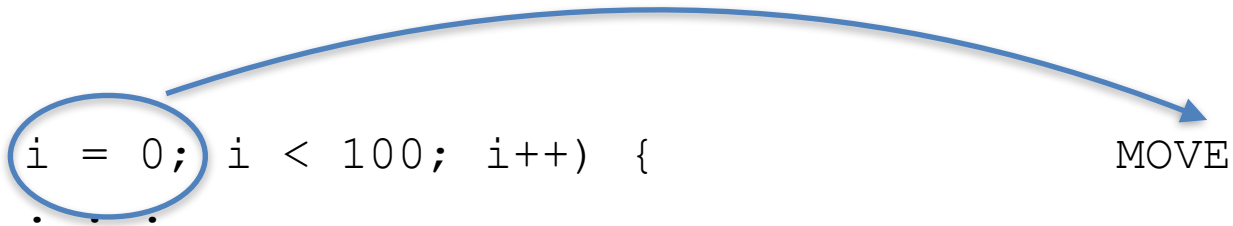
```
int i = 5;
```

```
for i = 0; i < 100; i++) {
    . . .
}
```

ASSEMBLY

```
MOVE.L #5,D0
```

```
MOVE.L #0,D0
```



DIRECT

Java

```
int i = 5;
```

ASSEMBLY

```
MOVE.L #5,D0
MOVE.L D0,I
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	00	00	00	00

I

DIRECT

Java

```
int i = 5;
```

ASSEMBLY

```
MOVE.L #5,D0
MOVE.L D0,I
```

DIRECT

```
LOOP:
     ADD.L #1, COUNT
     CMPI.L #15, COUNT
     BLT LOOP


COUNT:
    DC.L 10
```

COUNT 

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	00	00	00	0C

REGISTER INDIRECT

Java

```
int i = 5;
```

ASSEMBLY

```
MOVE.L #5,D0
LEA I,A0
MOVE.L D0,(A0)
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	00	00	00	00

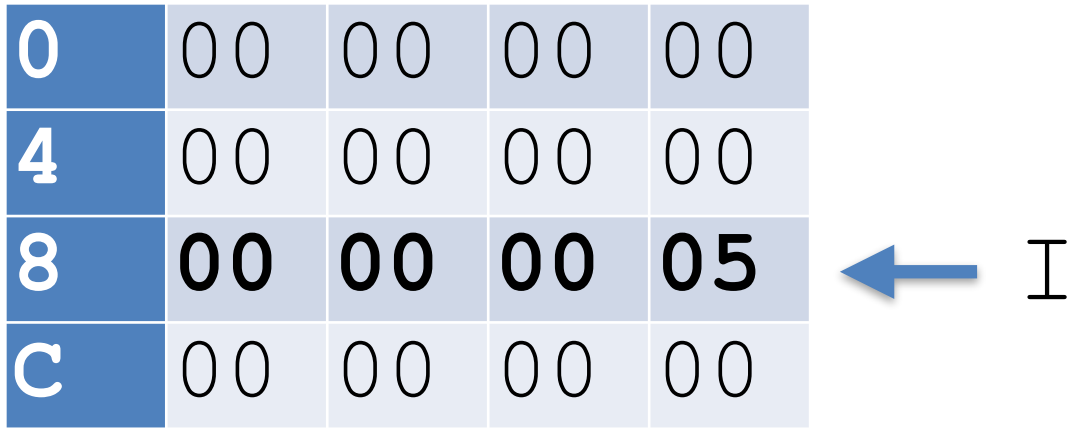
REGISTER INDIRECT

Java

```
int i = 5;
```

ASSEMBLY

```
MOVE.L #5,D0
LEA I,A0
MOVE.L D0,(A0)
```



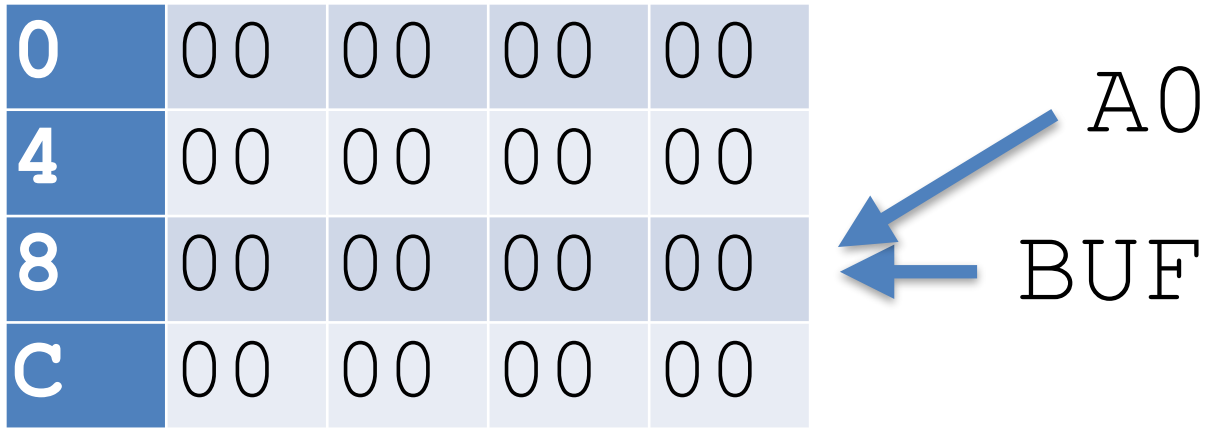
REGISTER INDIRECT

Java

```
char buf[10];
buf[0] = 0xff;
```

ASSEMBLY

```
LEA BUF,A0
MOVE.B #$FF,(A0)
```



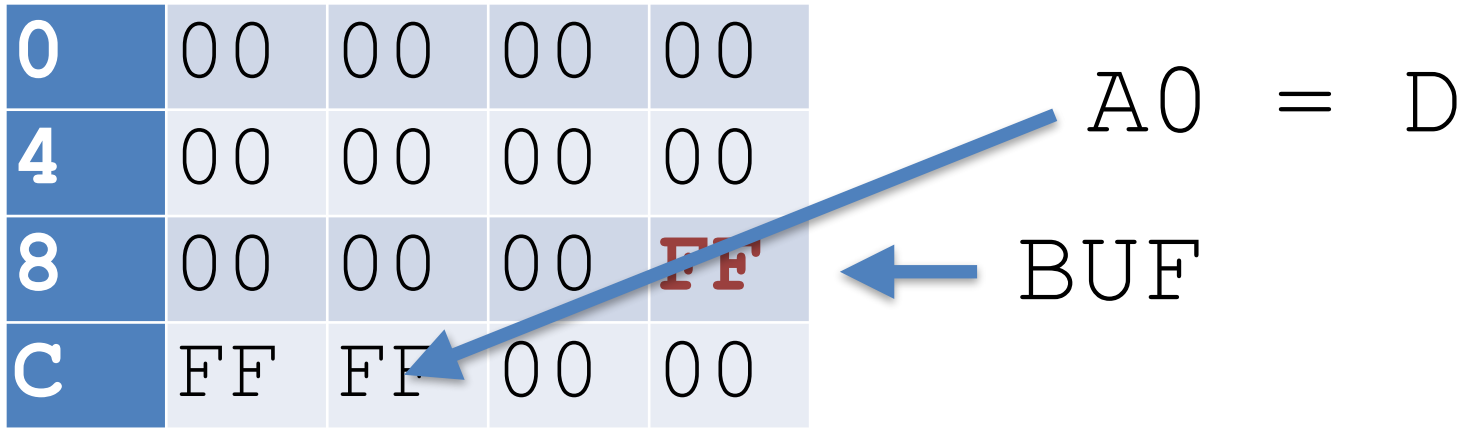
REGISTER INDIRECT

Java

```
char buf[10];
buf[0] = 0xff;
```

ASSEMBLY

```
LEA BUF,A0
MOVE.B #$FF,(A0)
```



REGISTER INDIRECT WITH INDEX

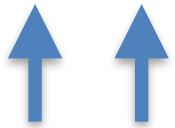
Java

```
char buf[10];
for(int k = 0; k < 10; k++){
    buf[0] = 0xff;
}
```

ASSEMBLY

```
LEA BUF,A0
MOVE.L #0,D0
MOVE.B #$FF,(A0,D0)
ADDI.L #1,D0
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	00	00	00	00



BUF A0+D0

REGISTER INDIRECT WITH INDEX

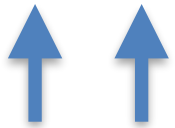
Java

```
char buf[10];
for(int k = 0; k < 10; k++){
    buf[0] = 0xff;
}
```

ASSEMBLY

```
LEA BUF,A0
MOVE.L #0,D0
MOVE.B #$FF,(A0,D0)
ADDI.L #1,D0
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	**FF**	00	00	00



BUF A0+D0

REGISTER INDIRECT WITH INDEX

Java

```
char buf[10];
for(int k = 0; k < 10; k++){
    buf[0] = 0xff;
}
```

ASSEMBLY

```
LEA BUF,A0
MOVE.L #0,D0
MOVE.B #$FF,(A0,D0)
ADDI.L #1,D0
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	FF	**FF**	00	00

BUF A0+D0

REGISTER INDIRECT WITH INDEX

Java

```
char buf[10];
for(int k = 0; k < 10; k++){
    buf[0] = 0xff;
}
```

ASSEMBLY

```
LEA BUF,A0
MOVE.L #0,D0
MOVE.B #$FF,(A0,D0)
ADDI.L #1,D0
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	FF	FF	**FF**	00

BUF A0+D0

REGISTER INDIRECT WITH POSTINCREMENT

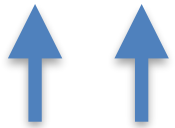
Java

```
char buf[10];
for(int k = 0; k < 10; k++){
    buf[0] = 0xff;
}
```

ASSEMBLY

```
LEA BUF,A0
MOVE.L #0,D0
MOVE.B #$FF,(A0)+
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	00	00	00	00



BUF A0

REGISTER INDIRECT WITH POSTINCREMENT

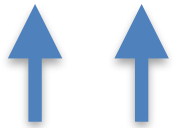
Java

```
char buf[10];
for(int k = 0; k < 10; k++){
    buf[0] = 0xff;
}
```

ASSEMBLY

```
LEA BUF,A0
MOVE.L #0,D0
MOVE.B #$FF,(A0)+
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	**FF**	00	00	00



BUF A0

REGISTER INDIRECT WITH POSTINCREMENT

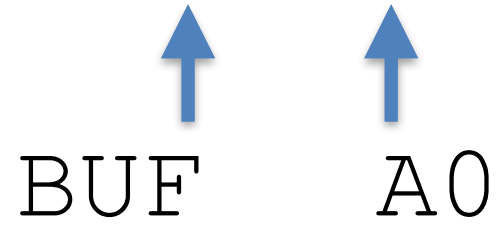
Java

```
char buf[10];
for(int k = 0; k < 10; k++){
    buf[0] = 0xff;
}
```

ASSEMBLY

```
LEA BUF,A0
MOVE.L #0,D0
MOVE.B #$FF,(A0)+
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	FF	**FF**	00	00



REGISTER INDIRECT WITH POSTINCREMENT

Java

```
char buf[10];
for(int k = 0; k < 10; k++){
    buf[0] = 0xff;
}
```

ASSEMBLY

```
LEA BUF,A0
MOVE.L #0,D0
MOVE.B #$FF,(A0)+
```

0	00	00	00	00
4	00	00	00	00
8	00	00	00	00
C	FF	FF	**FF**	00